ABSTRACT

Title of dissertation: APPLICATIONS OF GRAPH SEGMENTATION ALGORITHMS FOR QUANTITATIVE GENOMIC ANALYSES

Mohamed K. Gunady
Doctor of Philosophy, 2020

Dissertation directed by: Associate Professor, Héctor Corrada Bravo
Department of Computer Science

There is a growing interest in utilizing graph formulations and graph-based algorithms in different subproblems of genomic analysis. Since graphs provide a natural and efficient representation of sequences of data where some structural relationships are observed within the data, we study some graph applications in quantitative analysis of typical RNA sequencing (RNA-seq) and Whole Genome Sequencing pipelines.

Analysis of differential alternative splicing from RNA-seq data is complicated by the fact that many RNA-seq reads map to multiple transcripts, besides, the annotated transcripts are often a small subset of the possible transcripts of a gene. This work describes *Yanagi*, a tool for segmenting transcriptomes to create a library of maximal L-disjoint segments from a complete transcriptome annotation. That segment library preserves transcriptome substrings and structural relationships between transcripts while eliminating unnecessary sequence duplications.

First, we formalize the concept of transcriptome segmentation and propose an

efficient algorithm for generating segment libraries. The resulting segment sequences can be used with pseudo-alignment tools to quantify gene expression and alternative splicing at the segment level and provide gene-level visualization of the segments for more interpretability. The notion of transcript segmentation as introduced here and implemented in Yanagi opens the door for the application of lightweight, ultra-fast pseudo-alignment algorithms in a wide variety of RNA-seq analyses.

Furthermore, we show how transcriptome quantification can be performed from segment-level statistics. We present an EM algorithm that uses segment counts as features to estimate transcripts relative abundances in a way that maximizes the likelihood of the observed sequenced data. Then we tackle the problem of quantification in an incomplete annotation setting. We propose an assembly-free correction procedure that reduces bias in the estimated abundances of the annotated transcripts caused by the presence of unannotated transcripts in an RNA-seq sample, while avoiding the need to assemble the missing transcripts first.

Another use case of our graph segmentation approach is representing population reference genome graphs used in Whole Genome Sequencing (WGS), which can be crucial for some genomic analysis studying highly polymorphic genes like HLA. Usually graph-based aligners are slow and computationally demanding. Using segments empowers any linear aligner with the efficient graph representation of population variations, while avoiding the expensive computational overhead of aligning over graphs.

Lastly, we explore the use of Generative Adversarial Networks (GANs) for imputing the sparse and noisy expression data obtained in single cell RNA sequencing

(scRNA-seq) experiments. scRNA-seq provides a rich view into the heterogeneity underlying a cell population which is usually lost when performing bulk RNA-seq. However, these datasets are usually noisy and very sparse, and a number of methods have been proposed to impute zeros in these datasets with the goal of improving downstream analysis. In this work, we propose an approach, *scGAIN*, to impute zero counts of dropout genes in single cell data using Generative Adversarial Networks (GANs) by learning an approximation of the data distribution. The work presented here discusses an approach to adopt GAIN, a GAN model developed to impute data in image data, into the domain of imputing single cell data. Experiments show that scGAIN gives competitive results compared to the state-of-the-art imputation approaches while showing superiority in various aspects in simulation and real data. Imputation by scGAIN successfully recovers the underlying clustering of cell sub-populations, provides sharp estimates around true mean expression, reducing variability in the data, and increases the correspondence with matched bulk RNA-seq experiments.

APPLICATIONS OF GRAPH SEGMENTATION ALGORITHMS FOR QUANTITATIVE GENOMIC ANALYSES

by

Mohamed Khaled Geunady

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2020

Advisory Committee:
Professor Héctor Corrada Bravo, Chair/Advisor
Professor Steve M. Mount
Professor Mihai Pop
Professor Rob Patro
Professor Michael P. Cummings

Dedication

For Maha, Jannah and Fares!

For all healthcare workers who sacrifice a lot during pandemics!

For those who disembark into darkness hoping to create something different,

it is a worthwhile adventure!

Acknowledgments

I owe my utmost gratitude to *Allah*, the almighty, for giving me the power and strength to go through this journey and reach its finish line. I cannot thank Him properly without acknowledging all the people who made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost, I'd like to thank my advisor, Professor Héctor Corrada Bravo, for giving me the invaluable opportunity to be his student over the past few years. I will forever appreciate his time and effort he invested in me, always having open doors for help and guidance especially during my lowest moments. He kept me pushing towards developing my knowledge and working on interesting projects while always encouraging healthy work-life balance. It was an absolute honor to be mentored by such an extraordinary individual.

I would also like to thank Professor Steve Mount who inspired the main line of work done in this thesis. Without his motivating ideas and biological expertise, this thesis would have been a distant dream. Thanks are due to Professor Rob Patro for the fruitful collaboration. I have always enjoyed his engaging and insightful discussions. Thanks are also due to Professor Mihai Pop and Professor Michael Cummings for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript.

For my parents I owe the most gratitude for their infinite encouragement and prayers. They were always my pillars in life and the reason why I am here at this point. In addition, I would like to acknowledge my wife and partner, Maha. Your

continued support, love and patience is lightening up my life and made this dream possible. Thank you for standing by me through ease and hardship.

I take this as an opportunity to mention my colleagues at the Bravo lab, and our extended family at CBCB, for enriching my graduate life and building an environment full of fun and collaborations. Thanks to Jayaram Kancherla for helping and sharing his wide knowledge in various technical matters. Furthermore, for always having his room doors, literally, open for everyone. It was our mecca where we relax and exchange interesting thoughts with each other. My interaction with Justin Wagner, Aya Abdelsalam, Domenick Braccia, Theresa Alexander, Noor Singh, Yifan Yang, and Hirak Sarkar molded many of my thoughts, in research and life alike.

I would like to thank my community and friends in the house of Tauba for their friendship and support. They gave my family a sense of home far from home.

I would also like to acknowledge the help and support from some of the staff members. Barbara Lewis, Tom Hurst, Jennifer Story and Fatima Bangura always did their best to assist and coordinate logistics throughout the years. Without their coordination my graduate life would have been a mess.

Finally, I would like to acknowledge financial support from the National Institutes of Health (NIH), the National Science Foundation (NSF) and Illumina Inc for all the projects discussed herein.

It is impossible to mention all, and I apologize to those I've inadvertently left out.

Lastly, thank you all and thank God!

Table of Contents

List of Tables

List of Figures

List of Abbreviations

A3	Alternative 3' splice-site
A5	Alternative 5' splice-site
AF	Alternative First Exon
AL	Alternative Last Exon
AS	Alternative Splicing
CSG	contiguous Splice Graph
CV	Coefficient of Variation
DEU	Differential Exon Usage
DGE	Differential Gene Expression
DTU	Differential Transcript Usage
EC	Equivalence Class
EM	Expectation-Maximization
FDR	False Discovery Rate
GANs	Generative Adverserial Networks
HLA	Human Leukocyte Aantigen
INDEL	Insertion/Deletion
MSA	Multiple Sequence Alignment
MSE	Mean Squared Error
mRNA	Messenger RNA
MX	Mutually Exclusive Exons
PBMC	Peripheral Blood Mononuclear Cell
PCR	Polymerase Chain Reaction
PSI	Percent Spliced-In
RI	Retained Intron
RNA-seq	RNA Sequencing
SC	Segment Count
scRNA-seq	Single Cell RNA Sequencing
SE	Skipped Exon
SgG	Segments Graph
SNP	Single-Nucleotide Polymorphism
TCC	Transcripts Compatibility Count
TPM	Transcripts Per Million
t-SNE	t-Distributed Stochastic Neighbor Embedding

Chapter 1: Introduction

1.1 Overview

Advances in research that studies human genomes to understand biological systems in health and disease continue to grow. Research in that area faces significant challenges not only from the fact that many of the problems in the area are not well defined, with many latent factors that may play major roles in the study assumptions and hypotheses, but also an inherent challenge in the tremendous amount of data to be processed for many of the high throughput assays required to address these problems.

With the rising age of *BigData*, some research efforts became dedicated to developing algorithms and tools that are lightweight, ultra-fast and efficiently implemented, capable of processing the huge amount of data available. The need for both lightweight and accurate technologies is essential to remove bottlenecks in the analysis pipelines. In addition, there is a growing interest in utilizing graph formulations and graph-based algorithms in different subproblems of bioinformatics. Since graphs usually provide natural and efficient representation of sequences of data where some structural relationships are observed within the data, we explore some graph applications in quantitative genomics. In addition, we plan to use some of the

graph features obtained into prediction models that uses state-of-the-art machine learning approaches

1.2 Lightweight Approaches for Analyzing Short RNA-Seq Reads

Over the years, various approaches have addressed the joint problems of (gene level) transcript expression quantification and differential alternative RNA processing. Much effort in the area has been dedicated to the problem of efficient alignment, or pseudo-alignment, of reads to a genome or a transcriptome, since this is typically a bottleneck in the analytical processes that start with RNA-Seq reads and yield gene-level expression or differentially expressed transcripts. Among these approaches are alignment techniques such as bowtie [1], Tophat [2, 3], and Cufflinks [4], and newer techniques such as sailfish [5], RapMap [6], Kallisto [7] and Salmon [8], which provide efficient strategies through k-mer counting that are much faster, but maintain comparable, or superior, accuracy.

These methods simplified the expected outcome of the alignment step to find only the sufficient read-alignment information required by the quantification step. Given a transcriptome reference, an index of k-mers is created and used to find a mapping between reads and the list of compatible transcripts based on each approach's definition of compatibility. The next step, quantification, would be to resolve the ambiguity in reads that were mapped to multiple transcripts. Multi-mapping reads are common even assuming error free reads, due to shared regions produced by alternative splicing. The ambiguity in mapping reads is resolved using

probabilistic models, such as the EM algorithm, to produce the abundance estimate of each transcript [9]. It is at this step that transcript-level abundance estimation still faces substantial challenges that inherently affect the underlying analysis.

The presence of sequence repeats and paralogous genes in many organisms creates ambiguity in the placement of reads. More importantly, the fact that alternatively spliced isoforms share substantial portions of their coding regions, greatly increases the proportion of reads coming from these shared regions and consequently reads being multi-mapped becomes more frequent when aligning to annotated transcripts (Figure 2.1 A-B). In fact, local splicing variations can be joined combinatorically to create a very large number of possible transcripts from many genes. An extreme case is the Drosophila gene *Dscam*, which can produce over 38,000 transcripts by joining less than 50 exons [10]. More generally, long-read sequencing indicates that although there are correlations between distant splicing choices [11], a large number of possible combinations is typical. Thus, standard annotations, which enumerate only a minimal subset of transcripts from a gene (e.g. [12]) are inadequate descriptions. Furthermore, short read sequencing, which is likely to remain the norm for some time, does not provide information for long-range correlations between splicing events.

1.3 Gene and Transcript Expression Analyses

The standard RNA-seq pipeline for gene expression analysis depends on performing k-mer based alignment over the transcriptome to obtain transcripts abun-

dances, e.g. transcripts per million (TPMs). Then depending on the objective of the differential analysis, an appropriate hypothesis test is used to detect genes that are differentially expressed. Methods that perform differential gene expression (DGE) prepares gene abundances by summing the underlying transcript abundances. Consequently, DGE methods aims at testing for differences in the overall gene expression. Among these methods are: DESeq2 [13] and edgeR [14]. Such methods fail to detect cases where some transcripts switch usage levels while the total gene abundance is not significantly changing. Note that estimating gene abundances by summing counts from the underlying transcripts can be problematic, as discussed in [15]. RATs [16] on the other hand is among those methods that target to capture such behavior and tests for differential transcript usage (DTU). Regardless of the testing objective, both tests entirely depend on the transcript abundances that were obtained from algorithms like EM during the quantification step to resolve the ambiguity of the multi-mapped reads, which requires some bias-correction modeling ([8,17]) adding another layer of complexity to achieve the final goal of gene analysis.

1.4 Alternative Splicing Analysis

Within a gene, the study of how certain genomic regions are alternatively spliced into different isoforms is related to the study of relative transcript abundances. Each local splicing event describes a possible variation of splicing of the described genomic region. For instance, an exon cassette event (exon skipping) describes either including or excluding an exon between the upstream and downstream

exons. Consequently, isoforms are formed through a sequential combination of local splicing events. For binary events, the relative abundance of an event is commonly described in terms of percent spliced-in (PSI) [18] which measures the proportion of reads sequenced from one splicing possibility versus the alternative splicing possibility, while ΔPSI describes the difference in PSI across experimental conditions of interest.

Several approaches were introduced to study alternative splicing and its impact in studying multiple diseases. [19] surveyed eight different approaches that are commonly used in the area. These approaches can be roughly categorized into two categories depending on how the event abundance is derived for the analysis. The first category is considered count-based where the approach focuses on local measures spanning specific counting bins (e.g. exons or junctions) defining the event, like DEXSeq [20], MATS [21] and MAJIQ [22]. Unfortunately, many of these approaches can be expensive in terms of computation and/or storage requirements since it requires mapping reads to the genome, and then processing the huge matrix of counting bins. The second category is isoform-based where the approach uses the relative transcript abundances as basis to derive PSI values. This direction utilizes the transcript abundance (e.g. TPMs) as a summary of the behavior of the underlying local events. Cufflinks [4, 15], DiffSplice [23] and SUPPA [24, 25] are of that category. Unlike Cufflinks and DiffSplice which perform read assembly and discovers novel events, SUPPA succeeds in overcoming the computational and storage limitations by using transcript abundances that were rapidly prepared by lightweight k-mer counting alignment like Kallisto [7] or Salmon [8].

A main drawback of SUPPA and other transcript-based approaches alike is that it assumes a homogeneous abundance behavior across the transcript making it prone to coverage biases. Previous work showed that RNA-seq data suffers from coverage bias that needs to be modeled into methods that estimate transcript abundances [17, 26]. Sources of bias can vary between fragment length, positional bias due to RNA degradation, and GC content in the fragment sequences. Consider the diagram in figure 1.1 with a case of two isoforms where isoform1 has higher abundance than isoform2. Both isoforms involve two exon skipping events (E1, E2). The diagram shows the read coverage over different regions of both isoforms with exon E1 in particular has low relative coverage. Considering the real evidence of reads supporting the first skipping event E1, gives an incorrect conclusion when the overall abundances of the two isoforms involved is considered. More importantly, transcript-based approaches fail to provide different measures of confidence for differential analysis of events E1 and E2 since both events will have the same PSI values, whereas there is a significant difference in coverage supporting both events.

1.5 Population Genome Reference

For next-generation sequencing (NGS), the task of read alignment and extraction is crucial to most downstream genome analyses. Linear aligners depend on a genome reference that usually represent a single consensus haplotype, or a small set of individual contig variations in case of alt-aware aligners [27]. With the sheer diversity in genome sequences among individuals in many organisms like humans,

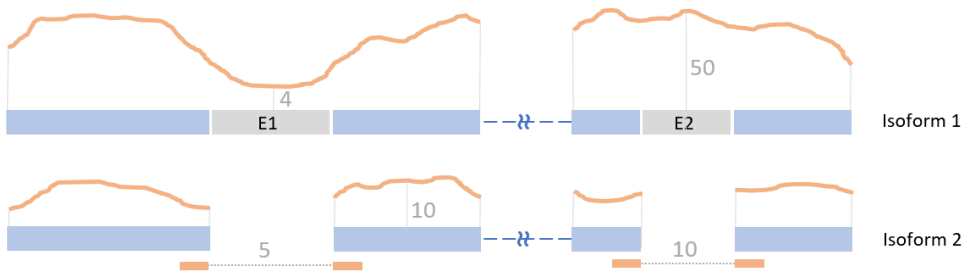

Figure 1.1: Diagram illustrates the coverage bias problem with AS transcript-based approaches. Given the two given isoforms where isoform1 has higher abundance than isoform2. The diagram shows the read coverage over different regions of both isoforms with exon E1 in particular has low relative coverage. Using the overall transcript abundances gives $PSI > 0.5$ for the first skipping event E1, whereas using the read evidence of the event gives $PSI < 0.5$. Additionally, using transcript abundances gives equal PSI values for both events E1 and E2 without any measure of confidence corresponding to their actual evidence.

the need to consider these characterized variations become inevitable to maintain accuracy in targeted analyses like genotyping and disease phenotyping.

A typical genome reference is a monoploid representation of the average genome during the assembly step. For example, a typical human genome differs from the reference genome at around 5.0 million sites [28]. Most sites encounter single nucleotide polymorphisms (SNPs), short insertions or deletions (INDELs), and significantly less frequent yet as important structural variants (SVs), with the actual number of variants highly varies depending on the individual's population [28]. Some genes are especially rich with such variants like the highly polymorphic human leukocyte antigen (HLA) genes. HLA genes encodes proteins that play a crucial role in the immune system and many disease phenotypes, in particular, the HLA Class I (HLA-A, -B, -C) and Class II (HLA-DQA1, -DQB1, -DRB1) genes, with over 3000 variants seen in HLA-B alone [29]. Studying such genes is of high medical importance but depending only on the reference genome for mapping sequenced reads is challeng-

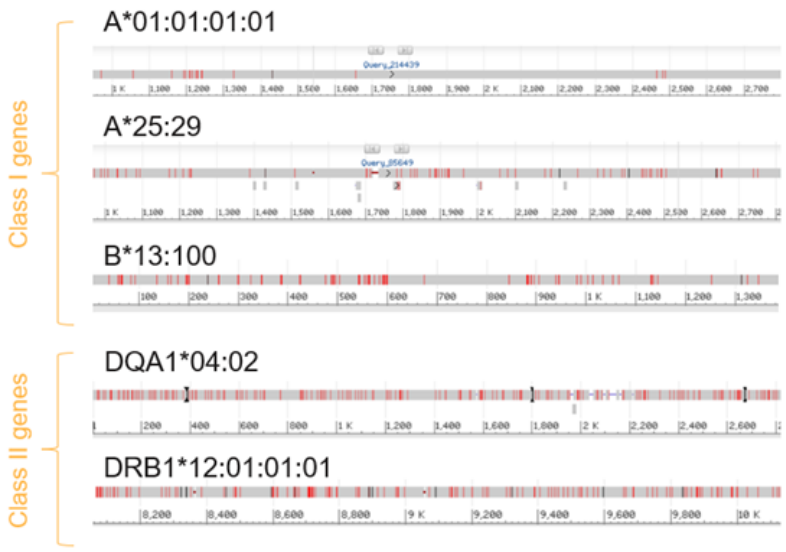

Figure 1.2: Examples of known HLA alleles. Three alleles from Class I genes and two alleles of Class II genes. Red marks show genomic variants from the reference sequence.

ing since a significant number of the reads are expected to have many mismatches leading to low quality mapping or a significant loss of reads. This challenge makes using a rather richer reference inevitable.

Since GRCH37 [30] attempts have been made to add several alternative scaffolds of some of the highly variable regions in the genome along with the primary assembly into the reference. Now the current human reference (GRCH38.p12) has 261 alternate sequences covering 178 regions. However, not only this approach is limited in the amount of data it provides, but also this approach fails at providing homology relationships among the alternative sequences. It is very likely to have multiple scaffolds share a major portion of their sequences, which leads to another challenge where several reads can be multi-mapped.

Some recent projects are devoted into building more comprehensive catalogs of

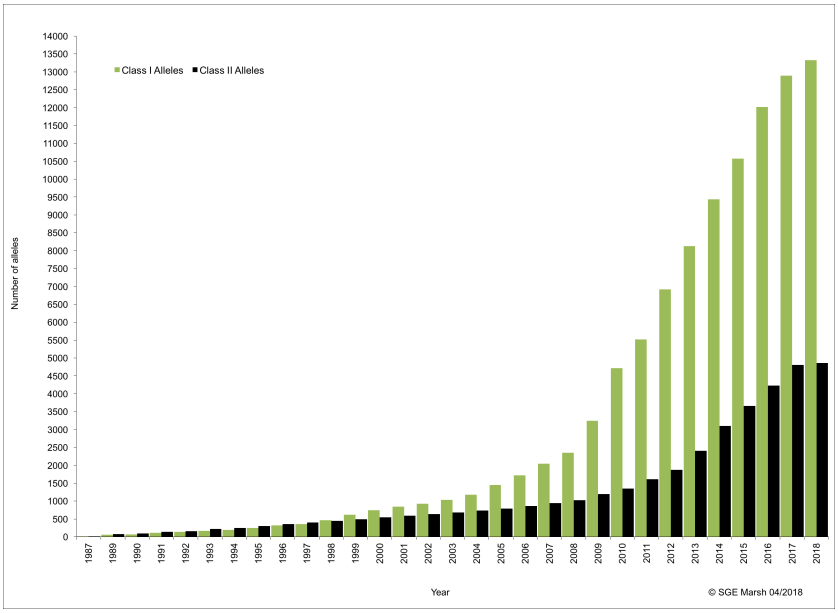

Figure 1.3: Yearly growth of IPD-IMGT/HLA database.

known genomic variants and providing it publicly. E.g. IPD-IMGT/HLA Database [29] provides up to 4000 alleles for each class I gene, and up to 2000 alleles for class II genes. Other projects like the 1000 Genomes Project [28,31] can be used to derive similar information for other regions of the human genome. Although such archives are rapidly growing, approaches in the field that efficiently utilize the available data are scarce.

Consider the hypothetical example in figure 1.4 with a set of alleles, each has a different set of variants. The sequence colors in the left panel shows how much of the sequences is shared. So even without the presence of structural variants, it is apparent that the alternative alleles share a common structure which branches over the possible variations at the corresponding loci, hence a graph can be a reasonable and efficient representation of these alleles. That observation ignited a growing interest into developing graph representations of such rich references. Hence, more research

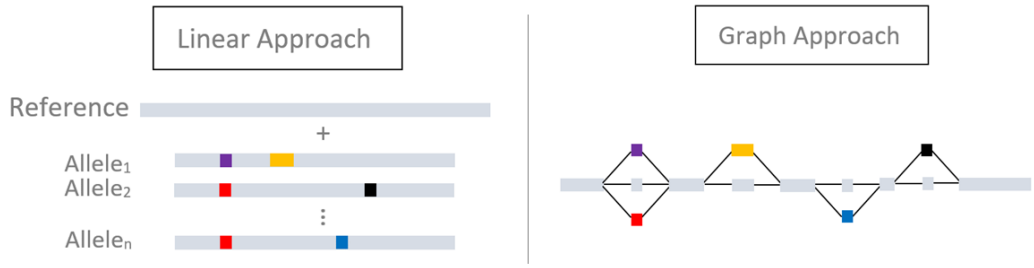

Figure 1.4: Two directions to incorporate alleles into alignment. (Left) Linear Approach: appends known alleles to the reference as decoy sequences. (Right) Graph Approach: builds a graph of the reference and known alleles and perform alignment over the graph.

now targets building graph-based aligners [32–34] either specifically over HLA genes or the whole genome in general. Paten et. al. provides a review of the different approaches and tools to build genome graphs [35]. Unfortunately, graph-based aligners are not mature yet facing a few challenges ahead. The lack of efficient implementations and algorithms make graph-based approaches very computationally expensive for practical application.

1.6 Thesis Contributions

The work discussed in this thesis presents a line of research that addresses the use of graph representation and algorithms in the kinds of genomic analyses discussed above. The benefits of our work vary from providing faster and more efficient pipelines, empowering available approaches with more capabilities or introducing new perspectives into existing challenges. We summarize the contributions of our work in the following points:

- Introducing the concept of transcriptome segmentation and formulating the definitions of segments and segment counts as summarized statistics of RNA-

seq experiments that preserve a localized measure across the genome.

- Illustrating how our framework can be used to run analyses on the three resolutions of RNA-seq: i.e. Gene-level, Transcript-level and Alternative Splicing-level analyses.

- Empowering lightweight, ultra-fast pseudo-alignment tools like Kallisto or Salmon with capabilities to provide alternative splicing measures that achieves comparable accuracy to count-based approaches while maintaining the speed and efficiency of such tools.

- Tackling the problem of transcript quantification under incomplete annotation and propose a segment-based correction procedure to reduce the bias in the estimated abundances present in such scenarios, without the need to assemble the missing transcripts first.

- Introducing *Yanagi* an open source tool that provides an efficient RNA-seq workflow where we implement most ideas present here.

- Bridging the gap between linear and graph alignment over whole genome population references and evaluating its benefits on handling allelic variations of highly polymorphic genes like HLA genes.

- Proposing a generative model approach to impute single cell RNA-seq data to correctly recover the missing values of dropout genes by adopting deep GANs for that purpose.

Chapter 2: Yanagi: Transcript Segment Library Construction for RNA-seq Quantification

2.1 Overview

Messenger RNA transcript abundance estimation from RNA-Seq data is a crucial task in high-throughput studies that seek to describe the effect of genetic or environmental changes on gene expression. Transcript-level analysis and abundance estimation can play a vital role in performing fine-grained analysis studying local splicing events and coarse-grained analysis studying changes in gene expressions.

In this work, we propose a novel strategy that aims at constructing a set of transcriptome segments that can be used in the read-alignment-quantification steps instead of the whole transcriptome without loss of information. Such a set of segments (a segment library) can fully describe individual events (primarily local splicing variation, but also editing sites or sequence variants) independently, leaving the estimation of transcript abundances as a separate problem. Here we introduce and formalize the idea of transcriptome segmentation, propose and analyze an algorithm for transcriptome segmentation, through a tool called Yanagi. To show how the segments library can be used in downstream analysis, we show results of using

Yanagi for gene-level and alternative splicing differential analysis.

2.2 Transcriptome Segmentation

Figure 2.1 shows a typical situation in RNA-seq data analysis and provides an overview of the transcript segmentation strategy. In particular, it summarizes how reads that would be multi-mapped when aligning to a transcript library would be aligned to segments. In the latter case, all reads are aligned to a single target sequence and read counts are obtained per segment without the need of probabilistic quantification methods to resolve ambiguity. The next few subsections present a few more specifics of the method for transcriptome segmentation in Yanagi [36, 37].

2.2.1 Segments Properties

Yanagi's objective is to generate a minimal set of disjoint sequences (where disjointness is parameterized by the experimental sequencing read length) while maintaining transcriptome sequence completeness.

The following definitions are for a given transcriptome T, and parameter L.

Definition 1 *A Segment*

A segment seg defined by the tuple $\langle exs, loc, w \rangle$ *is a genomic region of width* w *beginning at genomic location* loc *and spanning the sequence of consecutive exonic regions* $exs \in Exs_T$ *(either exons or retained introns). Exonic regions are considered consecutive if they are consecutively spliced into at least one possible isoform in T. The Width* w *of each segment in a segments library* $S_{T,L}$ *is at least* L *bases.*

Definition 2 *Segments Sequences Completeness*

The set of segments $S_{T,L}$ is Complete if and only if

$$seq \in S_{T,L}; \forall seq \in Substring(T), len(seq) \leq L$$

and

$$seq \in Substring(T); \forall seq \in Substring(S_{T,L})$$

Definition 3 *L-disjoint Segments*

Each segment in the set $S_{T,L}$ is L-disjoint if and only if

$$width[overlap(seg_i, seg_j)] < L; \forall seg_i, seg_j \in S, i \neq j$$

The L-disjointness property restricts any pair of *L-disjoint* segments to have an overlap region shorter than parameter L, which typically equals to the sequencing read length. In other words, no read of length at least L can be mapped to both segments of an *L-disjoint* segment pair, assuming error-free reads.

Another property of the generated segments is to be maximal. For $seg : \langle exs, loc, w \rangle$, denote $Txs(seg)$ as the set intersection of annotated transcripts splicing exons exs. We can define a subsumption relationship between segments as $seg_1 \succ seg_2$ if and only if $exs_1 = exs_2, loc_1 = loc_2, Txs(seg_1) = Txs(seg_2)$ and $w_1 > w_2$. With this relationship we can define the following property of a segment library $S_{T,L}$

Definition 4 *Maximal Segments*

For each segment in the set $S_{T,L}$ to be Maximal

$$seg_1 \succ seg_2 \Rightarrow seg_2 \notin S_{T,L}, \forall seg_1 \in S_{T,L}$$

Thus, a maximal segment is the longest common sequence of genomic regions starting at *loc*, such that these regions are spliced similarly, i.e. the entire sequence belongs to the same set of transcripts. That is why in figure 2.1 (C) segment S5 is extended to include two exons and their junction, while segment S2 is interrupted by the different splicings of Tx1 and Tx2.

2.3 Yanagi's Segmentation Algorithm

Given the transcriptome annotation (GTF format file) and the transcript sequences (FASTA format files) as input, Yanagi generates the set of segments and its sequences (as a FASTA file) as the output of the segmentation process. Figure 2.1 (F) illustrates an example of how Yanagi perform transcriptome segmentation given the splicing graph of a complex AS event studied in [22]. Recall, that in splicing graphs, nodes represent genomic regions and edges represent how the regions are spliced, while paths represent possible transcripts.

The transcriptome segmentation process can be summarized into three steps: (1) Preprocessing the transcriptome annotation to obtain disjoint exonic bins, (2) Constructing a Segments Graph (SgG), and finally (3) Generating the segment library. Transactions in Figure 2.1 (F) represent these three steps.

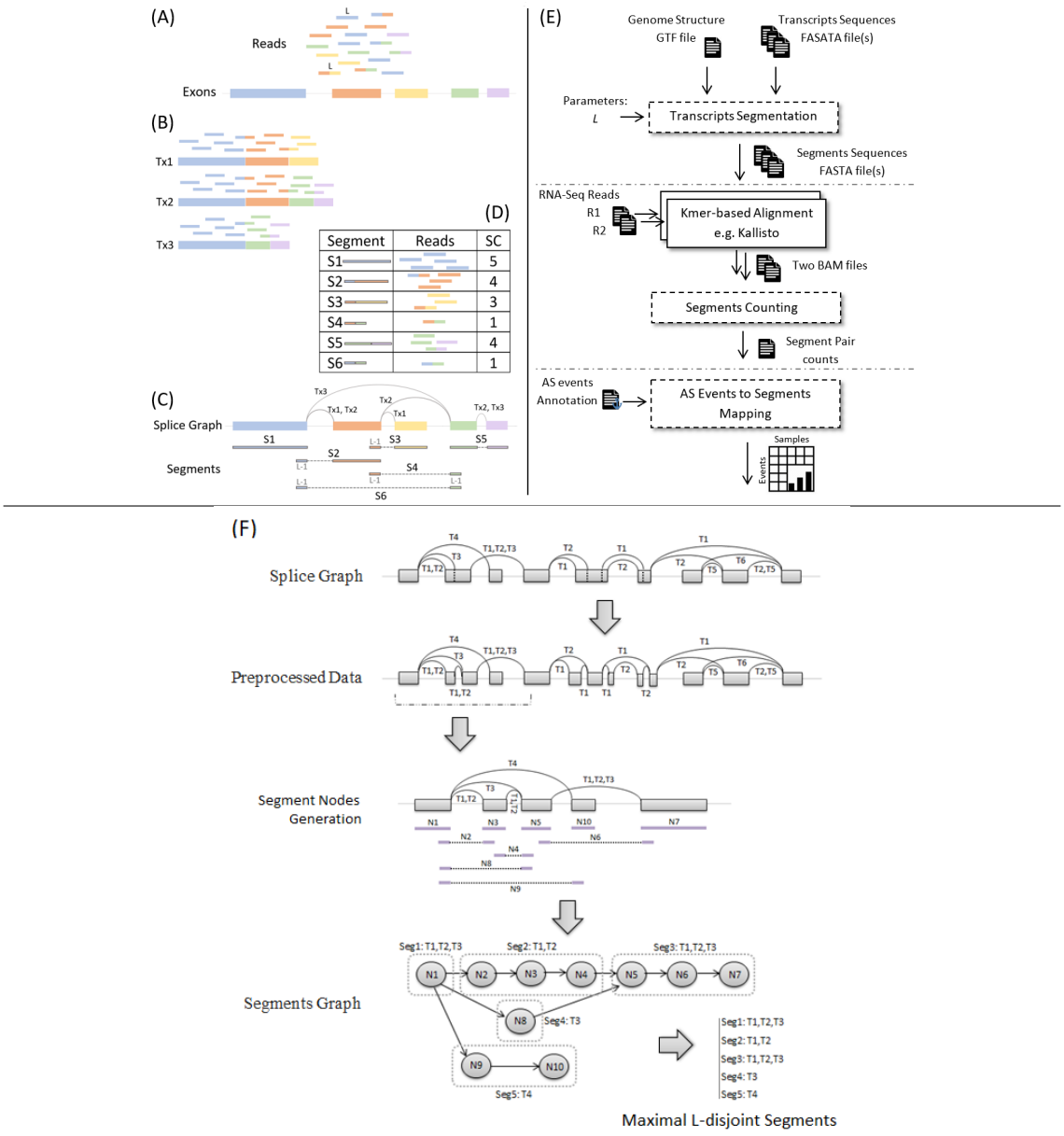

Figure 2.1: **An overview of transcriptome segmentation and Yanagi-based workflow.** The leftside shows a typical RNA-seq example along with the output from Yanagi. **(A)** Shows the example set of exons and its corresponding sequenced reads. **(B)** shows the result of alignment over the annotated three isoforms spliced from the exons. **(C)** shows the splice graph representation of the three isoforms along with the generated segments from yanagi. **(D)** shows the alignment outcome when using the segments, and its segment counts (SCs). **(E)** Yanagi-based workflow: segments are used to align a paired-end sample then use the segments counts for downstream alternative splicing analysis. Dotted blocks are components of Yanagi. **(F)** Three steps for generating segments starting from the splice graph for an example of a complex splicing event. Assuming no short exons for simplicity. Step two and three are cropped to include only the beginning portion of the graph for brevity.

Figure 2.2: **An example of naive segmentation based on exons and junctions.** Two cases are shown, each is represented using a splicing graph of two transcripts, along with a set of possible RNA-seq reads and the generated segments following the naive approach. **(Left)** The first case shows a simple case where the naive approach successfully generates segments spanning all possible reads. **(Right)** The second case shows a case of two short exons (E2, E3 of width $k < L$) where the naive approach fails to span the given read.

2.3.1 1. Annotation Preprocessing

In our algorithm, exons and junctions serve as initial candidates for segment generation. We apply a preprocessing step to eliminate exon overlaps present in the transcriptome reference from events involving alternative 3'/5' splice sites, or transcription start/end sites. This step ensures that any splicing event is occurring either at the beginning or the end of a genomic segment, which makes the process of generating L-disjoint and max covering segments easier. The preprocessing step is independent from the parameter L, so it can be done only once per transcriptome reference. We implemented the preprocessing step based on the *GenomicRanges* package in R , specifically, the *disjoin* function, which takes less than a few seconds to run on the human genome.

2.3.2 2. Constructing Segments Graph

Currently Yanagi builds a separate segment graph for each gene, since there are no alternative splicing events between transcripts of different genes. However, future work may use segment graphs that connect different genes sharing regions of identical sequence length L or greater, but we have yet to address this.

Definition 5 *Segment Graph*

A segment graph $G_{T,L}$ is an acyclic directed graph defined by the pair (N, E), where N is a set of nodes representing segments, and E is the set of directed edges between the nodes. An edge $e : (n_i, n_j) \in E$ is created if the segment corresponding to node n_i directly precedes the segment corresponding to node n_j in some transcript.

Definition 6 *Segment Node*

A segment node n is a vertex in segment Graph G that represents a seed of an L-disjoint segment such that $w_n \geq L$. A maximal segment may consist of one or more consecutive segment nodes.

2.3.3 3. Generating Segments

While full details of the algorithm are given in Appendix B, here we present a high-level description. For a given gene, the algorithm iterates over the set of annotated transcripts in that gene. A cursor *loc* starting at the beginning of a transcript slides over the sequence of genomic regions forming that transcript. Given the current cursor location loc_n, a seed of the node n is initiated. Then a refinement

step, explained further in the next paragraph, is used to handle cases involving exons shorter than L. The segment node is then added to the graph with the key pair (exs_n, loc_n) as the node identifier, and the cursor loc is advanced to the new location. It should be noted that the out-degree of each segment node corresponds to the number of upcoming alternative splices. The final step is generating the actual segments. Any segment with an out-degree greater than one is a candidate start of a segment. Each possible path beginning at a start-segment node till the following start-segment node (or a leaf node) produces an output segment. See Figure 2.1 (F) and Appendix B for the full details of the segmentation algorithm. It is worth mentioning that a segment graph may look like a de Bruijn graph (DBG) that is commonly used in assembly problems. However, a path of nodes in DBG represents a sequence of k-mer components while a path of nodes in SgG represents a sequence of genomic regions spliced into an isoform. As a result, the DBG built from the list of transcripts and the DBG built from the list of segments should be identical, since both graphs represent the same sequences of nucleotides.

Back to the issue of short regions which raises the possibilities of generating segment nodes of length L spanning more than two exonic regions. Once the key pair (exs_n, loc_n) is determined, the node refinement step determines the extent of that node and how the cursor loc should be advanced in order to preserve the L-disjointness constraint. Figure 2.3 shows a diagram of how a segment node is refined. The logic behind the refinement step is aggregating the sequence of nodes expected to be created spanning the same set of regions exs_n; since it is guaranteed that there are no splicing events occurring between the start and end of region

Ex_n, the next necessary segment node would be the node spanning part of Ex_{n+1}. Consequently, the new location of the cursor *loc* would be the location where the first segment spanning Ex_{n+1} starts. That aggregation improves the time and space complexity of the algorithm as it reduces the number of generated nodes and avoids shredding the genome in dense areas of short exons which may impose a problem in the quantification step as discussed in next subsection. In fact, the refinement step ensures that for every distinct value of exs, there is a maximum of two segment nodes generated and that reduces the algorithm complexity by factor of L.

As an attempt to analyze the complexity of the algorithm, we can estimate a loose upper bound of the number of segment nodes N in G. Consider a gene with T_g transcripts and E_g disjoint genomic regions (obtained by the preprocessing step), where the maximum width of such a region is w_{max}. Recalling the property mentioned in the previous paragraph, that a maximum of two nodes can be generated for the same set of regions, a segmentation iteration for a transcript that spans $E_t \leq E_g$ regions can generate $1 < o(E_t - \left\lceil \frac{L}{w_{max}} \right\rceil) < o(E_t)$ segment nodes. That gives the upper bound of $o[\sum_{T_g}(E_t - \left\lceil \frac{L}{w_{max}} \right\rceil)]$ or $o[T_g.(E_g - \left\lceil \frac{L}{w_{max}} \right\rceil)]$ for N. That means the time and space complexity of the graph construction increases when using lower values of parameter L, or with organisms of longer and more complex transcriptome structure. Table 2.1 shows time and memory analysis for constructing the segments library for two organisms used in our later analysis. The results show that running Yanagi is an efficient and fast process that does not add a burden in terms of time and space requirements.

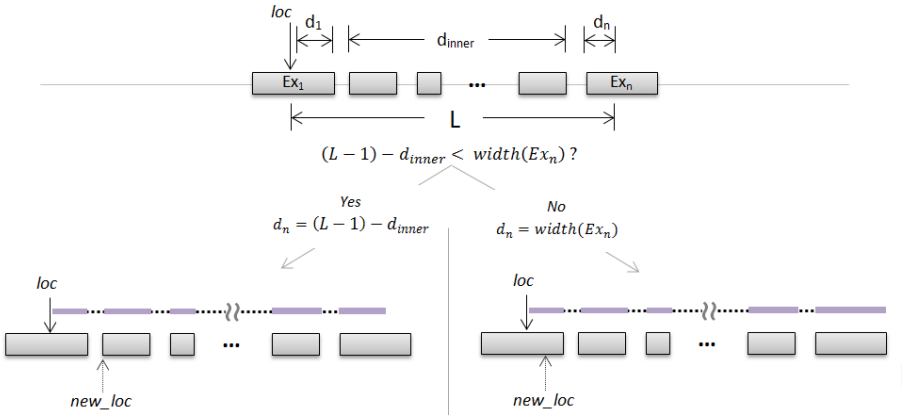

Figure 2.3: Diagram illustrates the node refinement step, for a node spanning n genomic regions. The step determines the extent of the node and how the cursor *loc* should advance in each of the two candidate cases.

Table 2.1: Running time (seconds) and memory usage (gigabytes) by Yanagi to generate segment library for fruit fly (Dm6) and human (Hg38) genomes, for both the preprocessing and segmentation steps. Time for the preprocessing step does not include the time to load the FASTA and GTF files. Most of the memory usage is from loading the input data in both steps. Running on a 6-core 2.1 GHz AMD processor, using single-threaded processes. The lower half shows the time and memory usage for running quasi-mapping step in RapMap [6] using either the segments library or the full transcriptome, to quantify samples of 40M paired-end reads, each of length 101bp.

	Dm6		Hg38	
	time(s)	memory(GB)	time(s)	memory(GB)
Preprocessing	13	0.9	112	1.5
Segmentation				
L=40	20	0.4	248	1.3
L=108	20	0.4	250	1.3
L=1000	20	0.4	228	1.3
L=10000	8.5	0.4	77	1.3
Rapmap Indexing				
L=108	103	0.8	420	2.6
Txs	121	1.1	480	3.7
Rapmap Quantification				
L=108	236	0.7	220	2.1
Txs	292	1.2	416	3.1

2.4 Yanagi-based Workflow

Figure 2.1 (E) gives an overview of a Yanagi-based workflow which consists of three steps. The first step is the transcriptome segmentation, in which the segments library is generated. Given the transcriptome annotation and the genome sequences, and for a specific parameter value L, Yanagi generates the segments in FASTA file format. This step of library preparation is done once independently from the samples. The second step is the alignment step. Using any k-mer based aligner e.g. kallisto or RapMap, the aligner uses the segments library for library indexing and alignment. The outcome of this step is read counts per segments (in case of single-end reads) or segment-pair counts (in case of paired-end reads). These segment counts (SCs) are the statistics that Yanagi provides to be used in any downstream analysis. The third step depends on the specific target analysis. Later on this work, we describe two use cases where using segment counts shows to be computationally efficient and statistically beneficial.

2.5 Analysis of Generated Segments

For practical understanding of the generated segments, we used Yanagi to build segment libraries for the fruit fly and human genomes: Drosophila melanogaster (UCSC dm6) and Homo sapiens (UCSC hg38) genome assemblies and annotations. These organisms show different genome characteristics, e.g. the fruit fly genome has longer exons and transcripts than the human genome, while the number of

transcripts per gene is much higher for human genome than the fruit fly. A summary of the properties of each genome is found in [38].

2.5.1 Sequence lengths of generated segments

Since L is the only parameter required by the segmentation algorithm, we tried different values of L to understand the impact of that choice on the generated segments library. Recall that the choice of L is based on the expected read length of the sequencing experiment. For this analysis we chose the set $L = (40, 100, 1000, 10000)$.

Figure 2.4 shows the histogram of the lengths of the generated segments compared to the histogram of the transcripts lengths, for each value of L, for both fruit fly (left) and human (right) genomes. The figure shows the expected behavior when increasing the value of L; using small values of L tends to shred the transcriptome more (higher frequencies for small sequence lengths), especially with genomes of complex splicing structure like the human genome. While with high values of L, such as $L = 10,000$, the minimum segment length anticipated tends to be higher than the length of most transcripts, ending up generating segments such that each segment represents a full transcript.

2.5.2 Number of generated segments per gene

Figure 2.5 shows how the number of generated segments in a gene is compared to the number of the transcripts in that gene, for each value of L, for both fruit fly (left) and human (right) genomes. A similar behavior is observed while increasing

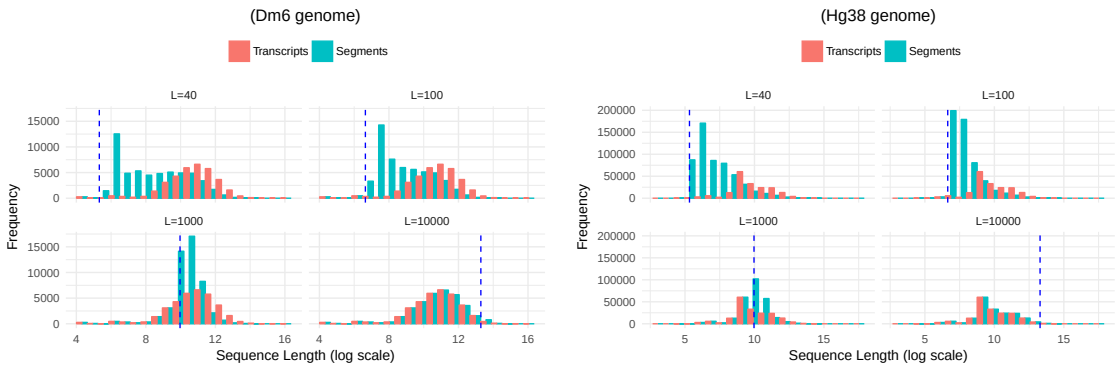

Figure 2.4: Histogram of transcripts lengths vs. segments lengths for both fruit fly (left) and human (right) genomes, with different values of L (40, 108, 1000, 10,000). Dotted vertical line represents the used value of L during the transcriptome segmentation.

the value L, as with the segment length distribution. The fitted line included in each scatter plot provides indication of how the number of target sequences grows compared to the original transcriptome. For example, when using $L = 100$ (a suitable value with Illumina's short reads), the number of target sequences per gene, which will be the target of the subsequent pseudo-alignment steps, almost doubles. It is clear from both figures the effect of the third step in the segmentation stage. It is important not to shred the transcriptome so much that the target sequences become very short leading to resulting complications in the pseudo-alignment and quantification steps, and not to increase the number of target sequences leading to increasing the processing complexity of these steps.

2.5.3 Library Size of the generated segments

As a summary, Table 2.2 shows the library size when using segments compared to the reference transcriptome in terms of the total number of sequences, sequence bases, and file sizes. The total number of sequence bases clearly shows the advan-

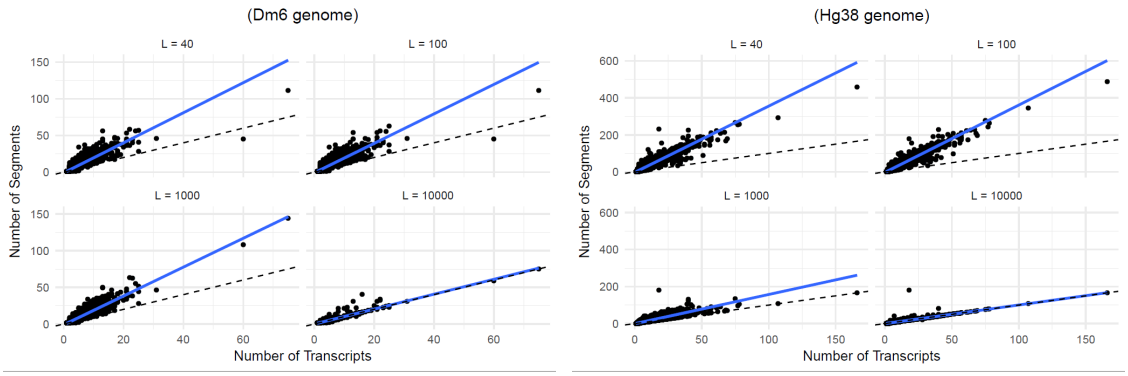

Figure 2.5: Number of transcripts vs. number of segments, per gene, for both fruit fly (left) and human (right) genomes, with different values of L (40, 108, 1000, 10,000). The figure shows how a fitted line (solid blue) compares to the identity line (dotted black).

tage of using segments to reduce repeated sequences appearing in the library that corresponds to genomic regions shared among multiple isoforms. For instance, using $L = 100$ achieves 54% and 35% compression rates in terms of sequence lengths for fruit-fly and human genomes, respectively. The higher the value of L is, the more overlap is allowed between segments, hence providing less the compression rate. Moreover, that necessarily hints on the expected behavior of the alignment step in terms of the frequency of multi-mappings.

Table 2.2: Library size summary of the generated segments for fruit fly (Dm6) and human (Hg38) genomes

	Txome	Segments			
		$L = 40$	$L = 100$	$L = 1000$	$L = 10000$
Dm6					
Number of bases (Gb)	90	39	41	71	90
Number of Sequences	34,681	54,680	53,694	48,741	34,625
FASTA File Size (MB)	89	44	47	76	92
Hg38					
Number of bases (Gb)	278	147	181	308	281
Number of Sequences	182,435	544,991	541,361	264,083	183,165
FASTA File Size (MB)	276	206	239	338	302

2.5.4 Impact of using segments on Multi-mapped Reads

To study the impact of using the segments library instead of the transcriptome for alignment, we created segments library with different values of L and observed the number of multi-mapped and unmapped reads for each case and how it is compared to when the transcriptome is used. We used RapMap [6] as our k-mer based aligner, to align samples of 40 million simulated reads of length 101 (samples from the dataset discussed in Datasets section) in a single-end mode. The experimented values of L were centered around the value of $L = 101$ with more value points close to 101 to test how sensitive the results are towards slight changes in the selection of L. Figure 2.6 shows the alignment performance in terms of the number of multi-mapped reads (red solid line) and unmapped reads (blue solid line), compared to the number of multi-mapped reads (red dotted line) and unmapped reads (blue dotted line) when aligning using the transcriptome. Using segments highly reduces the number of multi-mapped reads. The plot shows that too short segments compared to the read length results in a lot of unmapped reads. Consequently, choosing L to be close to the read length is the optimal choice to minimize multimappings while maintaining a steady number of mapped reads. It is important to note that the best segments configuration still produces some multimappings. That is a result of the presence of reads sequenced from paralogs and sequence repeats that are not tackled in the current version of Yanagi. However, using segments can achieve around 10-fold decrease in the number of multimappings.

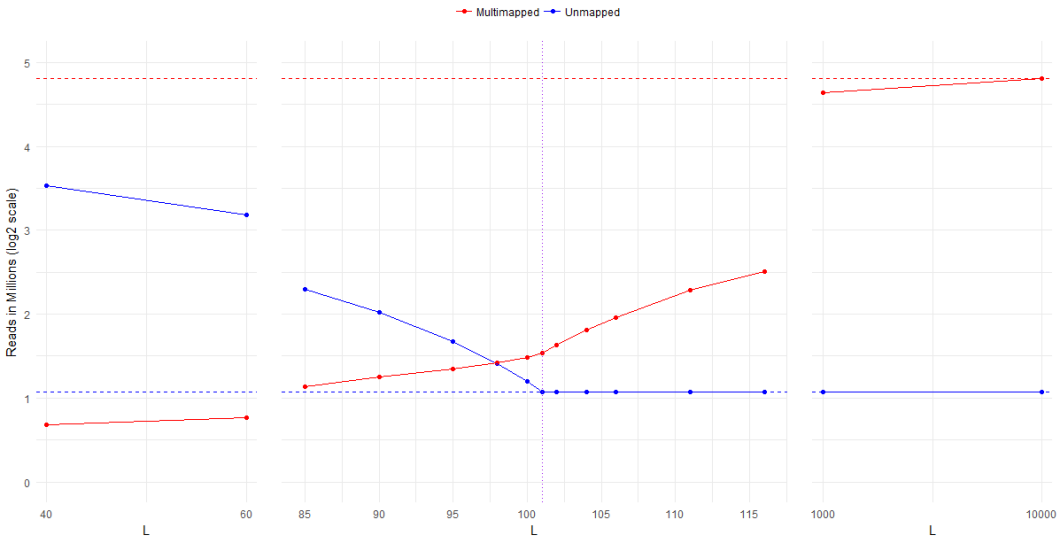

Figure 2.6: Alignment performance using Segments from hg37, tested for different values of L, to align 40 million reads of length 101 (first sample in simulated dataset 4.5. Performance is shown in terms of the number of multi-mapped reads (red solid line) and unmapped reads (blue solid line), compared to the number of multi-mapped reads (red dotted line) and unmapped reads (blue dotted line) when aligning using the transcriptome.

2.5.5 The importance of maximality property

Recalling that the generated segments are maximal segments, as mentioned in definition 2.2.1. It is the property that segments are extended as much as possible between branching points in the segments graph. The purpose of this property is to maintain stability in the produced segment counts; Since shorter segments will inherently produce lower counts which introduces higher variability that can complicate the downstream analysis. Figure 2.7 shows the distribution of coefficient of variation (CV) of the produced segment counts from segments with and without maximal property. To examine the effect of the maximal property, we simulated 10 replicates from 1000 random genes (with more than two isoforms) from the hg38 transcriptome using *Ployester* [39]. When segments are created without maximal

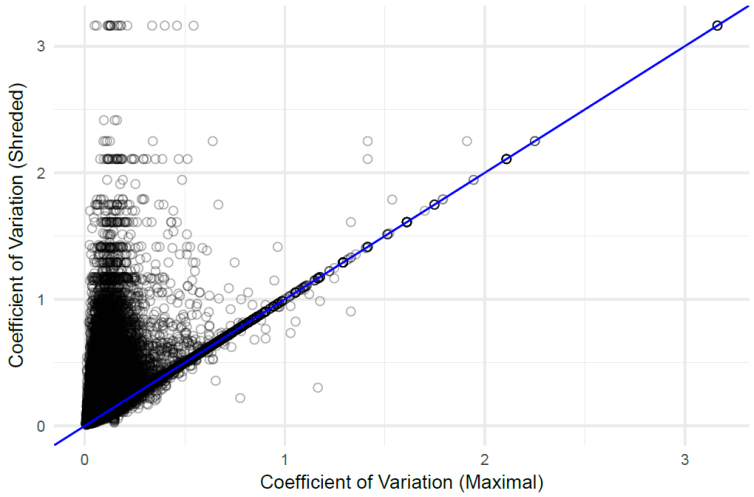

Figure 2.7: Distribution of coefficient of variation for segment counts produced from maximal segments versus segments without the maximal property enforced. Reads of 10 replicates are simulated from 1000 random genes (with more than two isoforms) from hg38 transcriptome.

property, the scatter plot clearly shows that maximal segments have lower CVs to their corresponding short segments for a majority of points (40% of the points has a difference in CVs > 0.05). That corresponds to generating counts with lower means and/or higher variances if the maximal property was dropped.

2.6 Discussion

In this chapter we introduce Yanagi, an efficient tool that creates disjoint segments of reference transcriptomes amenable for quantification of RNA-seq reads using pseudo-alignment techniques. We have formalized the notion of transcriptome segmentation and proposed an efficient algorithm for constructing L-disjoint, max-spanning segments. We report on the characteristics of segment libraries in Drosophila melanogaster and Homo sapiens and use the resulting segments in a use case of differential analysis of exon skipping events across samples in two conditions

of interest.

Although it may appear that the discussed Yanagi-based workflow can perform quantification only for the annotated transcripts, the workflow can be extended to discover unannotated transcripts. An unannotated junction can be detected during the segment quantification stage by relaxing the restriction of accepting alignments of a pair of reads only if the pair of segments belong to at least one annotated transcript. When this restriction is relaxed, an unannotated junction can be detected when reads show enough evidence of that junction. I.e. when a segment pair that has no transcripts in common has high enough count. That problem is further discussed in chapter 5.

Finally, the issues of paralogs and intersecting genes are not tackled in the scope of this chapter. However, it is clear that there is no extra alignment complexity added due to these issues over the transcriptome-based alignment. Consequently, the occurrence of multi-mapping resulting from such cases also remains the same as the transcriptome-based quantification. So, a warranted extension to the current approach of Yanagi is to consider distinct genes that share identical exonic regions of length greater than L altogether.

The concept of transcriptome segmentation, and a tool that can build a segments library, opens the door for more extended RNA-seq analyses which we will discuss in the next few chapters. For instance, segment counts can serve as statistics into algorithms for differential isoform usage analysis, for which existing pseudo-alignment methods are commonly used. Moreover, segment level quantification can provide much more flexible opportunities for analysis including quantification of

RNA editing or other non-splicing variations. Currently we are exploring the possibility of utilizing the concept of segmentation into the problem of variant calling.

Chapter 3: Segment-based Gene Expression Analyses

3.1 Overview

A typical segment-based approach to do gene expression analysis would start by performing k-mer based alignment over the segments library prepared earlier by Yanagi using high-throughput tools like Kallisto, Sailfish or RapMap, to derive segment counts (SCs). The segment counts are then used to perform differential gene expression.

Our segment-based approach aims at breaking the coupling between the quantification, bias modeling, and gene expression analysis, while maintaining the advantage of using ultra-fast pseudo-alignment techniques provided by k-mer based aligners. When Aligning over the L-disjoint segments, the problem of multimapping across target sequences is avoided and as a result the quantification step can be dropped. Then the hypothesis test for differences across conditions are performed on SCs count matrix instead of TPMs.

3.2 Gene Expression Analysis: Kallisto's TCC-based approach

Yi et al. introduces a comparable approach in [40]. This approach uses an intermediate set defined in Kallisto's index core as equivalence classes (ECs). Specifically, a set of k-mers are grouped into an equivalence class (EC) if it belongs to the same set of transcripts during the transcriptome reference indexing step. Then during the alignment step Kallisto derives a count statistic for each EC. The statistics are referred to as transcripts compatibility counts (TCCs). In other words, Kallisto produces one TCC per EC representing number of fragments that appeared compatible with the corresponding set of transcripts during the pseudo-alignment step. Then the work in [40] uses these TCCs to directly perform gene-level differential analysis by skipping the quantification step using logistic regression. We will refer to that direction as TCC-based approach. To put that approach into perspective with our segment-based approach, we will discuss how the two approaches are compared to each other.

3.3 Gene Expression Analysis: Comparison between segment-based and TCC-based approaches

Both segment-based and TCC-based approaches avoid the quantification step when targeting gene-level analysis. This is considered an advantage in efficiency, speed, simplicity, and accuracy, as previously discussed. One difference is that segment-based approach is agnostic to the alignment technique used, while TCC-

based approach is a Kallisto-specific approach. More importantly, the statistic used in segment-based approach is easily interpretable. Since segments are formed to preserve the genomic location and splicing structure of genes, SCs can be directly mapped and interpreted with respect to the genome coordinates. However, ECs do not have a direct biological meaning in this sense. For instance, all k-mers that belong to the same transcript yet originated from distinct locations over the genome will all fall under the same EC, making TCCs less interpretable. While on the contrary, these k-mers will appear in different segments depending on the transcriptome structure. This advantage can be crucial for a biologist who tries to interpret the outcome of the differential analysis. In the next section we show a segment-based gene visualization that allows users to visually examine, for genes determined to be differentially expressed, what transcripts, exons and splicing events contributed to that difference.

Figure 3.1-bottom shows the number of segments per gene in Yanagi versus the number of equivalence classes per gene in Kallisto. The number of equivalence classes were obtained by building Kallisto's index on hg37 transcriptome, then running the pseudo command of Kallisto (Kallisto 0.43) on the 6 simulated samples in the SwitchTx dataset (section 4.5).Note that, in principle there should be more segments than ECs since segments preserve localization, however in practice Kallisto reports more ECs than those discovered in the annotation alone in some genes. The extra ECs are formed during pesudo-alignment when reads show evidence of unannotated junctions.

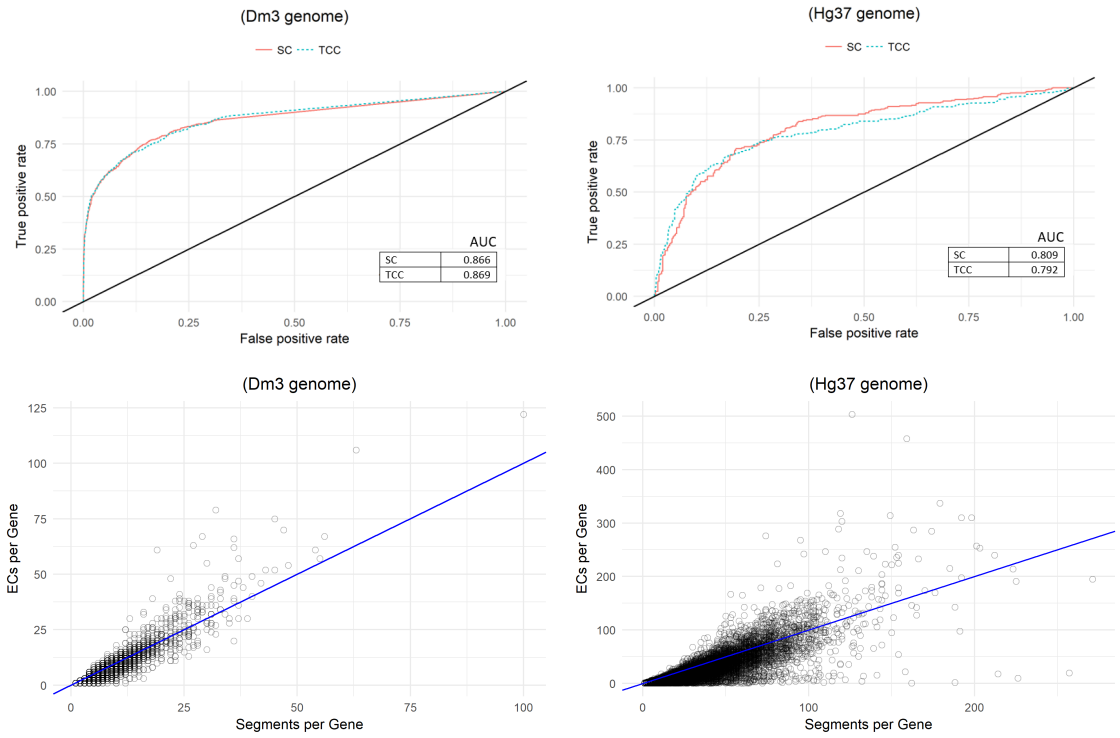

Figure 3.1: Segment-based gene-level differential expression analysis. **Top** ROC curve for simulation data for DEX-Seq based differential gene-level differential expression test based on segment counts (SC) and Kallisto equivalence class counts (TCC) for *D. melanogaster* and *H. sapiens*. **Bottom** Scatter plot of number of segments per gene (x-axis) vs. equivalence classes per gene in Kallisto (y-axis) for the same pair of transcriptomes.

3.4 Segment-based Gene Differential Expression Analysis

In this work we adopt the DEXSeq [20] method to perform segment-based gene differential analysis. DEXSeq is a method that performs differential exon usage (DEU). The standard DEXSeq workflow begins by aligning reads to a reference genome (not to the transcriptome) using TopHat2 or STAR [41] to derive exon counts. Then, given the exon counts matrix and the transcriptome annotation, DEXSeq tests for DEU after handling coverage biases, technical and biological variations. It fits, per gene, a negative binomial (NB) generalized linear model (GLM) accounting for effect of the condition factor and compares it to the null model (without the condition factor) using a chi-square test. Exons that have their null hypotheses rejected are identified as differentially expressed across conditions. DEXSeq can then produce a list of genes with at least one exon with significant differential usage and controls the false discovery rate (FDR) at the gene level using the Benjamini–Hochberg procedure.

We adopt the DEXSeq model for the case of segments by replacing exons counts with segments counts, the latter derived from pseudo-alignment. Once segments are tested for differential usage across conditions, the same procedure provided by DEXSeq is used to control FDR on the list of genes that showed at least one segment with significant differential usage.

We tested that model on simulated data (SwitchTx dataset in section 4.5)for both human and fruit fly samples and compared our segment-based approach with the TCC-based approach since they are closely comparable. Since the subject of

study is the effectiveness of using either SCs or TCCs as a statistic, we fed TCCs reported by Kallisto to DEXSeq's model as well to eliminate any performance bias due the testing model. As expected, figure 3.1-middle shows that both approaches provide highly comparable results on the tested dataset. Recall that using segment counts to test for differentially expressed genes adds to the interpretability of the test outcomes.

Although that experiment was chosen to test the use of SCs or TCCs as statistics to perform differential usage, different gene-level tests can also be performed on segment counts. For instance, testing for significant differences in overall gene expression is possible based on segment counts as well. A possible procedure for that purpose would be using DESeq2. One can prepare the abundance matrix by R package *tximport* [42], except that the matrix now represent segment instead of transcript abundances. The next section shows how visualizing segment counts connects the result of some hypotheses testing with the underlying biology of the gene.

3.5 Segment-based Gene Visualization

Figure 3.2 shows Yanagi's proposed method to visualize segments and the segment counts of a single gene with differentially expressed genes. The plot includes different panels combined, each showing a different aspect of the mechanisms involved in differential expression calls. The main panel of the plot is the segment-exon membership matrix (Panel A). This matrix plot shows the structure of the segments (rows) over the exonic bins (columns) prepared during the annotation

preprocessing step. Recall that an exon (or a retained intron) in the genome can be represented with more than one exonic bin in case of within-exon splicing events (Step 1 in section 2.3). Panel B is a transcript-exon membership matrix. It encapsulates the transcriptome annotation with transcripts as rows and the exonic bins as columns. Both membership matrices together allow the user to map segments (through exonic bins) to transcripts.

Panel C shows the segment counts (SCs) for each segment row. Panel D shows the length distribution of the exonic bins. Panel E is optional. It adds the transcript abundances of the samples, if provided. This can be useful to capture cases where coverage biases over the transcriptome is considered, or to capture local switching in abundances that are inconsistent with the overall abundances of the transcripts

The gene in figure 3.2 is on the reverse strand, that's why the exonic bins axis is reversed and segments are created from right to left. Consider segment S.0674 for instance. It was formed by spanning the first exonic bin (right-most bin) plus the junction between the first two bins. This junction is present only at transcript T.1354 and hence that segment belongs to only that transcript. In the segment-exon matrix, red-colored cells mean that the segment spans the entire bin, while salmon-colored cells represent partial bin spanning; usually at the start or end of a segment with correspondence to some junction.

Alternative splicing events can be easily visualized from figure 3.2. For instance, segments S.0672 and S.0671 represent an exon-skipping event where the exon is spliced in T.6733 and skipped in both T.1354 and T.9593.

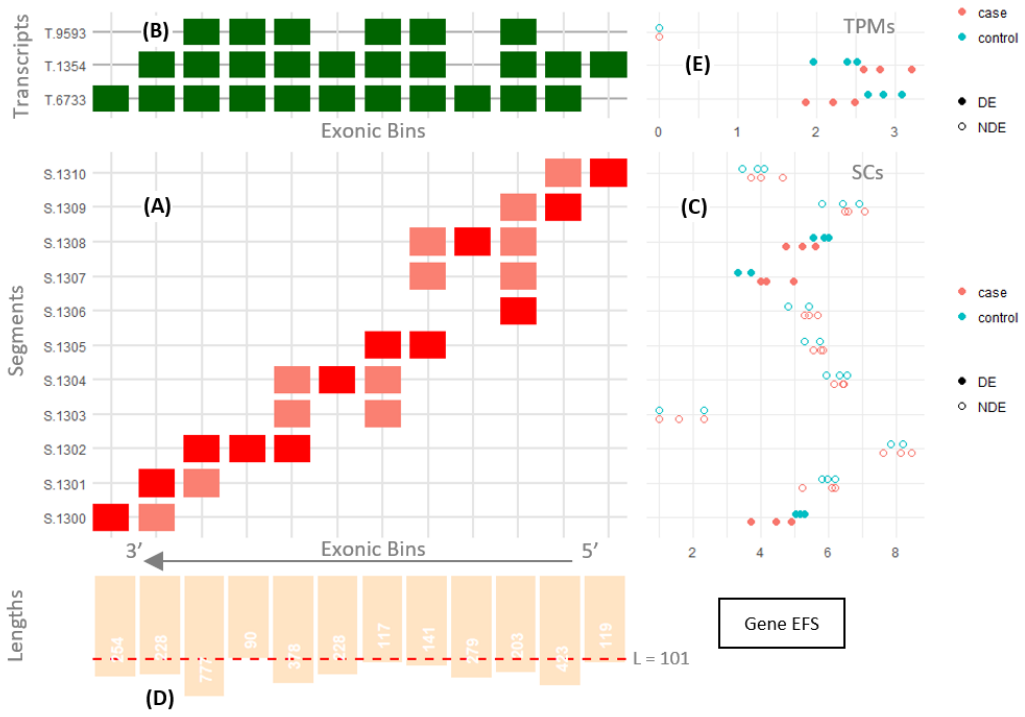

Figure 3.2: **Visualizing segments and segment counts of a single gene with differentially expressed transcripts.** It shows human gene **EFS** (Ensembl ENSG00000100842, genome build Hg37). The gene is on the reverse strand, so the bins axis is reversed, and segments are created from right to left. **(A)** Segment-exonic bin membership matrix, **(B)** Transcript-exonic bin membership matrix. **(C)** Segment counts for three control and three case samples, fill used to indicate segments that were significantly differential in the gene. **(D)** Segment length bar chart, **(E)** (optional) Estimated TPMs for each transcript.

.

3.6 Discussion

In this chapter, we illustrate how segment counts can be utilized as statistics to perform gene expression analysis and provide comparable performance to one of the state-of-the-art factorization techniques like Kallisto. Besides, we designed a gene visualization plot that summarizes the RNA-seq experiment to provide better interpretability of how the samples behave under that gene and provides insightful combination of analysis outcomes on the levels of genomic regions, transcripts and splicing events.

Chapter 4: Segment-based Alternative Splicing Analysis

4.1 Overview

Our segment-based approach works as a middle ground between count-based and transcript-based approaches. It provides local measures of splicing events while avoiding the computational and storage expenses of count-based approaches by using the rapid lightweight aligners that transcript-based approaches use. Our pipeline begins by running k-mer based lightweight alignment tools like Kallisto over the segments library prepared by Yanagi and obtain the segment counts. Yanagi's script is then used to map splicing events to their corresponding segments, e.g. each event is mapped into two sets of segments: The first set spans the inclusion splice, and the second for the alternative splice. Current version of Yanagi follows SUPPA's notation for defining a splice event and can process seven event types: Skipping Exon (SE), Retain Intron (RI), Mutually Exclusive Exons (MX), Alternative 5' splicec-site (A5), Alternative 3' splicec-site (A3), Alternative First Exon (AF) and Alternative Last Exon (AL).

Several approaches were introduced to study alternative splicing and its impact in studying multiple diseases. [19] surveyed eight different approaches that are commonly used in the area. These approaches can be roughly categorized into two

categories depending on how the event abundance is derived for the analysis. The first category is considered count-based where the approach focuses on local measures spanning specific counting bins (e.g. exons or junctions) defining the event, like DEXSeq [20], MATS [21] and MAJIQ [22]. Unfortunately, many of these approaches can be expensive in terms of computation and/or storage requirements since it requires mapping reads to the genome and subsequent processing of the large matrix of counting bins. The second category is isoform-based where the approach uses the relative transcript abundances as basis to derive PSI values. This direction utilizes the transcript abundance (e.g. TPMs) as a summary of the behavior of the underlying local events. Cufflinks [4,15], DiffSplice [23] and SUPPA [24,25] are of that category. Unlike Cufflinks and DiffSplice which perform read assembly and discovers novel events, SUPPA succeeds in overcoming the computational and storage limitations by using transcript abundances that were rapidly prepared by lightweight k-mer counting alignment like Kallisto or Salmon.

One drawback of SUPPA and other transcript-based approaches alike is that it assumes a homogeneous abundance behavior across the transcript making it susceptible to coverage biases. Previous work showed that RNA-seq data suffers from coverage bias that needs to be modeled into methods that estimate transcript abundances [17, 26]. Sources of bias can vary between fragment length, positional bias due to RNA degradation, and GC content in the fragment sequences.

Another critical drawback with transcript-based approaches is that its accuracy highly depends on the completeness of the transcript annotation. As mentioned earlier standard transcriptome annotations enumerate only a parsimonious subset of

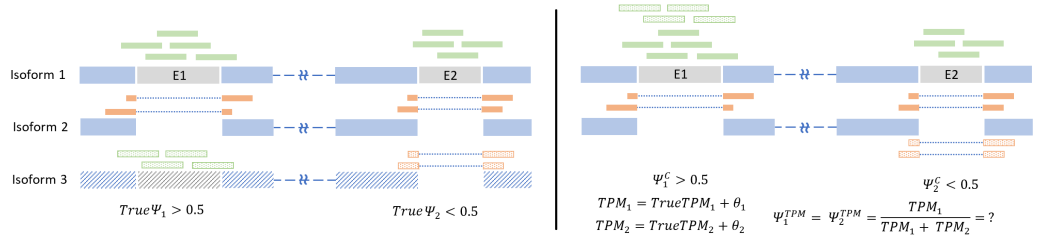

Figure 4.1: **Diagram illustrating a problem with transcript-based approaches for calculating PSI in the presence of unannotated transcripts. (Left)** shows the truth, with three isoforms combining two exon skipping events (E1, E2). However, isoform 3 is missing from the annotation. Reads spanning both events are shown along their true source. Reads spanning an exon incluion are colored green whereas reads spanning a skipping junction are colored orange. **(Right)** shows the problem with PSI values from transcript abundance. Because these two alternative splicing events are coupled in the annotation, their PSI values calculated from transcript abundances will always be the same ($\psi_1^{TPM} = \psi_2^{TPM}$), even though the true values are not ($True\psi_1 \neq True\psi_2$). Furthermore, changes in the estimated abundances (TPM_1, TPM_2) make the calculated PSI values unpredictable. Count-based PSI values (ψ_1^C, ψ_2^C) on the other hand correctly reflect the truth.

all possible sequential combinations of the present splicing events. Consider the diagram in figure 4.1 with a case of two annotated isoforms (Isoform 1 and 2) whereas a third isoform (isoform 3) is missing from the annotation. The three isoforms represent three possible combinations of two splicing events (skipping exons E1 and E2). If the two events are sufficiently far apart in genomic location, short reads would fail to provide evidence of the presence of isoform 3, leading to mis-assignment of reads into the other two isoforms (Figure 4.1 right). That behavior can bias the calculated PSI values of both events E1 and E2. Even if the mis-assigned reads did not change the estimation of TPM_1 and TPM_2, the calculated PSIs for both events can be significantly far from the truth. Further in this chapter we refer to any pair of events that involves such behavior as coupled events.

Our segment-based approach works as a middle ground between count-based

and transcript-based approaches. It provides local measures of splicing events while avoiding the computational and storage expenses of count-based approaches by using the rapid lightweight alignment strategies that transcript-based approaches use. Once the segment counts are prepared from the alignment step, Yanagi maps splicing events to their corresponding segments, e.g. each event is mapped into two sets of segments: The first set spans the inclusion splice, and the second for the alternative splice (See Methods 4.2). Current version of Yanagi follows SUPPA's notation for defining a splice event and can process seven event types: Skipping Exon (Skipped Exon (SE)), Retain Intron (Retained Intron (RI)), Mutually Exclusive Exons (Mutually Exclusive Exons (MX)), Alternative 5' splice-site (Alternative 5' splice-site (A5)), Alternative 3' splice-site (Alternative 3' splice-site (A3)), Alternative First Exon (AF) and Alternative Last Exon (AL).

4.2 Segment-based calculation of PSI

While Yanagi uses the transcriptome annotation to prepare the segments along with the splicing events, it generates mapping between each event and its corresponding segments spanning the event. For each event, Yanagi takes into consideration the transcripts involved and the event genomic coordinates to decide the set of transcriptome segments that correspond to each of the two possibilities of the splicing event. This step becomes complicated in case of overlapping events. The current version of Yanagi selects segments that spans either the event exon or junctions while the segment belong to at least one transcript that undergoes the

corresponding splicing.

After alignment, Yanagi provides segment counts or segment-pair counts in case of paired-end reads. For each splicing event, we calculate the PSI value of event e in sample x as follows:

$$PSI(e,x) = \frac{\sum_{s \in S_i(e)} SC(s,x)}{\sum_{s \in S_i(e) \cup S_e(e)} SC(s,x)}$$

where $S_i(e)$ and $S_e(e)$ are inclusion and exclusion segments, respectively, and $SC(s,x)$ is the segment count in the sample. That means segment-based PSI values uses reads spanning both the junctions and the target inclusion exon towards the inclusion count. In fact, read counts will also include reads extended around the event as long as the segment extends on both sides. This extension takes advantage of situations where splicing events are near to include as much discriminative reads into the counts to achieve higher levels of confidence when calculating PSI values.

4.2.1 Comparing Segment-based and isoform-based PSI values with incomplete annotation

To show how the estimated transcript abundances in the case of incomplete annotations can affect local splicing analysis, we ran both SUPPA and Yanagi pipelines on dataset simulating situations like the one in Figure 4.1. We simulated reads from 2454 genes of the human genome. A novel isoform is formed in each gene by combining two genomically distant events in the same gene (coupled events) where the inclusion of the first and the alternative splicing of the second does not appear in

any of the annotated isoforms of that gene (IncompTx dataset in section 4.5). After reads are simulated from the annotated plus novel isoforms, both SUPPA and Yanagi pipelines where run with the original annotation which does not contain the novel isoforms.

Figure 4.2 shows the calculated PSI values of the coupled events compared to the true PSI values. It is clear how the PSI values for both events can be severely affected by the biased estimated abundances. In SUPPA's case, abundance of both sets of inclusion and exclusion isoforms were overestimated. However, the error in abundance estimates of inclusion transcripts were consistently higher than the error in exclusion transcripts. Therefore, the PSI values of the second event were consistently overestimated by SUPPA whereas PSI values of the first events were consistently underestimated. Furthermore, splicing events involving the affected isoforms will be inherently affected as well even when they were unrelated to the missing transcript. This coupling problem between events inherent in transcript-based approaches is circumvented in values calculated by Yanagi, and generally, by count-based approaches.

Figure 4.3 shows the trends in estimation error of PSI across methods for the 2454 coupled events. ΔPSI of an event is calculated here as the difference between the calculated PSI of that event obtained either by Yanagi or SUPPA, and the true PSI. For each splicing event couple, a line connecting ΔPSI of the first event to the second's is drawn to show the trend of change in error between the first and second event in each pair. We found that estimates by SUPPA drastically exhibit a trend we refer to as overestimation-to-underestimation (or underestimation-to-

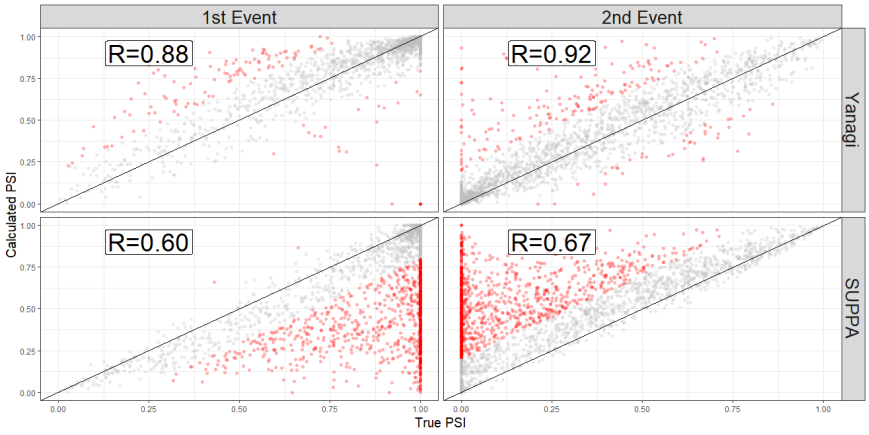

Figure 4.2: The PSI values of 2454 coupled events formulating novel isoforms used in data simulating scenarios of incomplete annotation, similar to figure 4.1. Each novel isoform consists of combining the inclusion splicing of the first event and the alternative (skipping) splicing of the second event. PSI values obtained by Yanagi and SUPPA are compared to the true PSI values. Red points are measures of error larger than 0.2. SUPPA tends to underestimate the PSI of the first event and overestimate in the second event (43% of the points are red compared to only 7% in Yanagi).

overestimation) in 50% of the pairs while 36% of the pairs showed minor errors ($\Delta PSI < 0.2$). Yanagi's estimates on the other hand showed the further trend only in 7% of the pairs while 87% of the pairs showed minor errors.

4.2.2 Comparing Segment-based and isoform-based PSI values on drosophila melanogaster

Based on known complexity and incompleteness of the *D*rosophila melanogaster transcript annotation we examined an RNA-seq dataset of male fly head (available online with GEO accession number GSM2108304) for evidence of similar behavior to that studied in the previous simulation. Since the true PSI values are unknown, we compare the trends of the difference in PSI between SUPPA and Yanagi. We

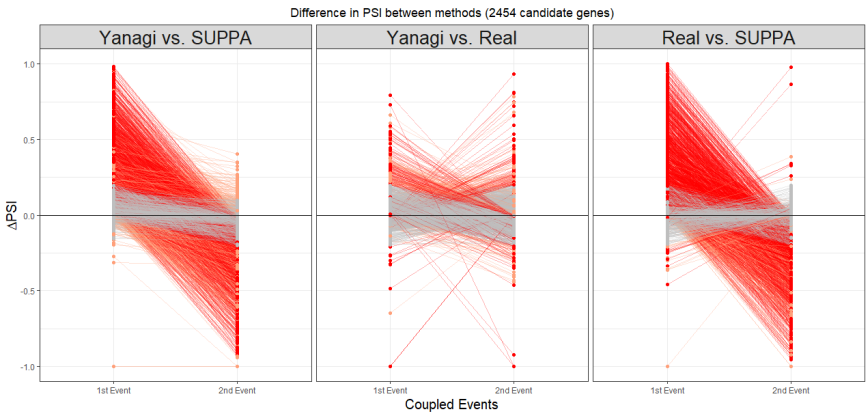

Figure 4.3: Trends of error in event PSI values across methods. ΔPSI of an event is calculated here as the difference in the calculated PSI of that event obtained either by Yanagi, SUPPA, or the truth. For each coupled event, a line connecting ΔPSI of the first event to the second's is drawn to show the trend of change in error among the first and second event in each pair. Overestimation-to-underestimation (and underestimation-to-overestimation) trends are colored red. Orange colored trends represent trends where both events were either overestimated or underestimated. Trends with insignificant differences ($|\Delta PSI| < 0.2$) are colored grey.

add to the comparison the PSIs obtained from a count-based approach, rMATS.

The scenario studied in the simulation is just one possible scenario of missing isoforms. More complex scenarios are likely to occur in real situations. Complex scenarios may include missing more than one isoform or when the event coupling problem involves more than two events. Such scenarios make detecting the full scale of the problem more complicated. Here we focus on the issue of coupled events as described in our simulation.

We follow the same analogy used in the simulation to define coupled events and find candidate genes of at least one missing isoform that couples two sufficiently distant events. By searching genes only in the forward strand and only events of type SE, A3, A5, we found 172 candidate genes and pair of coupled events where some splicing combination is possibly missing. Note that this candidate search is

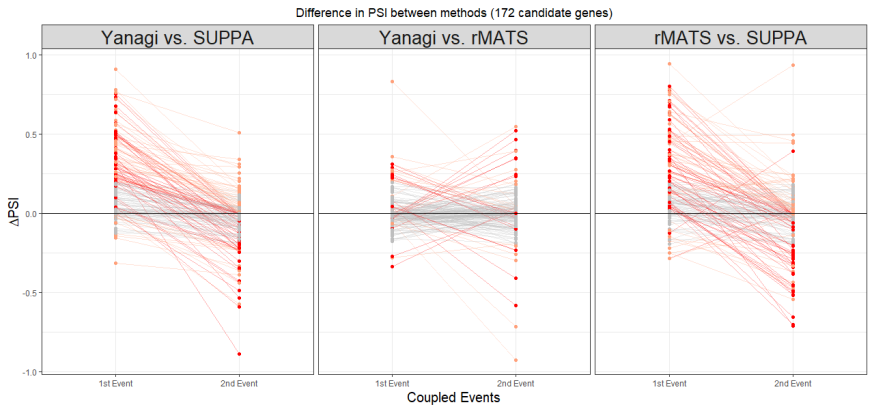

Figure 4.4: Trends in ΔPSI across methods Yanagi, SUPPA, rMATS for 172 coupled events in candidate genes for incomplete annotation in drosophila melanogaster (SRR3332174). Overestimation-to-underestimation (and underestimation-to-overestimation) trends are colored red. Orange colored trends represent trends where both events were either overestimated or underestimated. Trends with insignificant differences ($|\Delta PSI| < 0.2$) are colored grey. Out of the 172 cases, 33% showed Overestimation-to-underestimation (or underestimation-to-overestimation) trends in Yanagi-SUPPA, 11% in Yanagi-rMATS, 29% in rMATS-SUPPA.

independent of the RNA-seq data, or the segment generation process. Figure 4.4 shows the trends in ΔPSI between Yanagi, SUPPA and rMATS for the 172 cases of coupled events. Evidence of overestimation-to-underestimation trends were found between SUPPA and both Yanagi and rMATS, suggesting a similar behavior to the phenomenon present in our simulation (33% in Yanagi-SUPPA, 11% in Yanagi-rMATS, 29% in rMATS-SUPPA). It should be noted that those 172 cases of coupled events were only selected from part of the genome as candidates of one scenario of missing isoforms which means it is very likely for more cases to exist at the scale of the whole transcriptome. Figure 4.5 shows a scatter plot of the PSI values of full list of events found in the transcriptome annotation.

We study the Bruchpilot gene (FBgn0259246) as a specific illustration of a candidate gene with coupled events exhibiting overestimation-to-underestimation trend

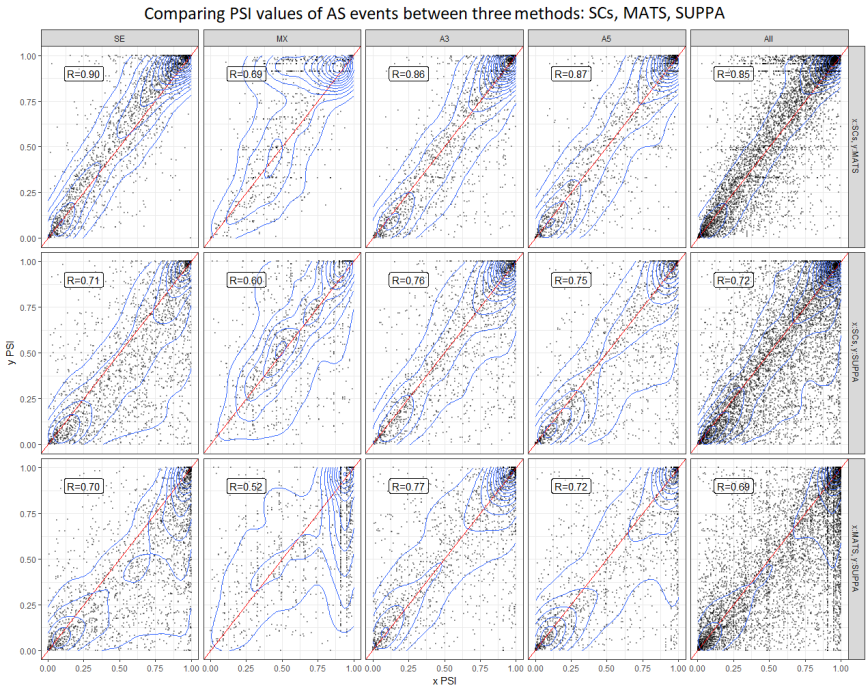

Figure 4.5: Comparing PSI values calculated using Yanagi (SCs), rMATS and SUPPA. Plots are stratified by event types on drosophila melanogaster sample (SRR3332174).

in SUPPA's ΔPSIs on Drosophila sample SRR3332174. Figure 4.6 shows three panels: (top panel) the read coverage of the genomic region of the gene by IGV alongside the 9 annotated transcripts, (bottom left panel) the segments visualization and its counts along with the transcripts abundances estimated by Kallisto, (bottom right panel) the PSI values of the coupled events E1, E2 calculated by SUPPA, Yanagi and rMATS. The read coverage for both events supports Yanagi's results rather than SUPPA's. The overestimation of one particular transcript, NM_001259298.2 (T.5059 in figure), can be one potential cause of such deviation. As the read coverage panel shows, most of the reads supporting that transcript are in fact coming from the first coding exon (its junction segment is highlighted grey) whereas the rest of the junctions, e.g. the skipping junction in E1, does not show sufficient coverage

supporting its high abundance estimated by Kallisto. One possible explanation is that the annotation is missing isoform X (colored green on the top panel). It is the same as the present transcript T.5059 except it combines the skipping splicing for E1 and the inclusion splicing for E2. The inclusion of isoform X in the annotation during transcript abundance estimation would have directed most reads aligned to the first exon towards isoform X rather than T.5059 for a more consistent coverage over both transcripts. Consequently, SUPPA's PSI values for both E1 and E2 would align better with Yanagi and rMATS values.

4.3 Comparing segment-based PSI values with counting-based and isoform-based PSI values

Here we are comparing PSI values obtained from Yanagi versus counting-based approaches like rMATS and isoform-based approaches like SUPPA on a very controlled setting. In that setting, we expect no significant difference between measures obtained from each of the three approaches. We used the simulation of switching abundance dataset (SwitchTx dataset in 4.5). Since each tool provides separate set of events, we focus our comparison on the intersection set of events between SUPPA and rMATS. That includes events from five types of splicing events. Table 4.1 summarizes the number of events subject to the study. Two levels of filtering are applied to observe how the different approaches behave in different scenarios. Non-overlapping events is the smallest subset of events. Those events exclude complex splicings where more than two splicings define the event. While highTPM events is

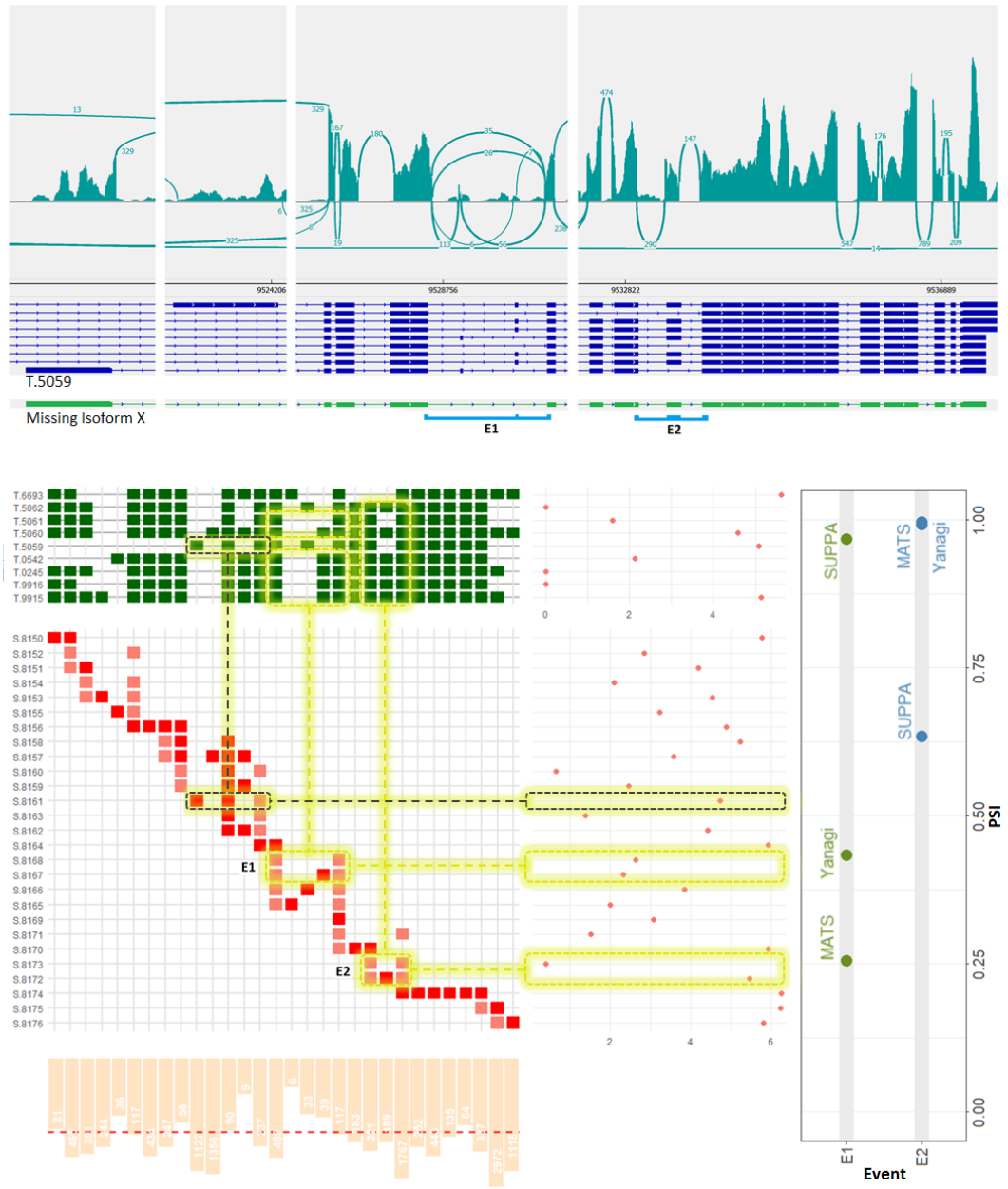

Figure 4.6: The *Bruchpilot* gene in *Drosophila melanogaster* (SRR3332174) serves as an example of a gene likely to have incomplete annotation. (**Bottom-Right**) The *PSI* values of the coupled events E1 and E2 exhibit severe overestimation and underestimation, respectively, by transcript-based approaches compared to Yanagi and rMATS. (**Top**) illustrates read coverage across the gene prepared using IGV, aligned with the 9 annotated isoforms. (**Bottom-Left**) The segments visualization of the gene is compared to transcript-level expression (TPM) obtained from kallisto, and the segment counts (normalized) from Yanagi's pipeline. Refer to section 3.5 for details on this panel's components. Postulating a isoform X (shown as a green-colored track on the top panel) missing from the annotation explains the deviation in both *PSI* values and the inconsistency in coverage across transcript T.5059.

a subset of events in which inclusion and exclusion isoform levels are relatively high ($TPM_{inc} > 1, TPM_{ex} > 1$). This is a typical filtering criterion adopted by isoform-based approaches. This filter excludes events involving isoforms of low levels of expression which inherently suffer from low estimation accuracy. Note that when complex events are included, they are treated as a set of separate binary events.

Table 4.1: Number of Events in GRCh37 common between MATS and SUPPA for the five event types reported by both tools. Two levels of filtering are applied to obtain three subsets. Non-overlapping events are the simplest events where there is no more splicing other than the two possibilities defining the event. While highTPM events are events where inclusion and exclusion isoform levels are relatively high ($TPM_{inc} > 1, TPM_{ex} > 1$).

Events Subset	SE	MX	A3	A5	RI	Total
Non-overlapping	4,180	68	1,435	885	323	6,891
HighTPM Events	9,756	354	2,327	1,483	793	14,713
All Events	13,650	1,024	3,131	2,053	1,711	21,569

Figure 4.7 (Top) shows a scatter plot of PSI values calculated by the three approaches for all events. Separate plots for the filtered events in figure 4.8. Among the five different splicing types exon skipping, alternative 3' and alternative 5' events give the highest correlation between segment counts and rMATS approaches. In our experiments we noticed that rMATS (v4.0.1) does not behave as intended for intron retention events. We noticed that counts including junction reads only and counts including both junction and intron reads (which we use in this study) are the same. In other words, rMATS fails to report reads spanning the intron, which explains the underestimated inclusion counts and PSI values for retained introns.

It should be noted that most count-based approaches require aligning to the genome which is usually the bottle-neck process in the pipeline that some try to over-

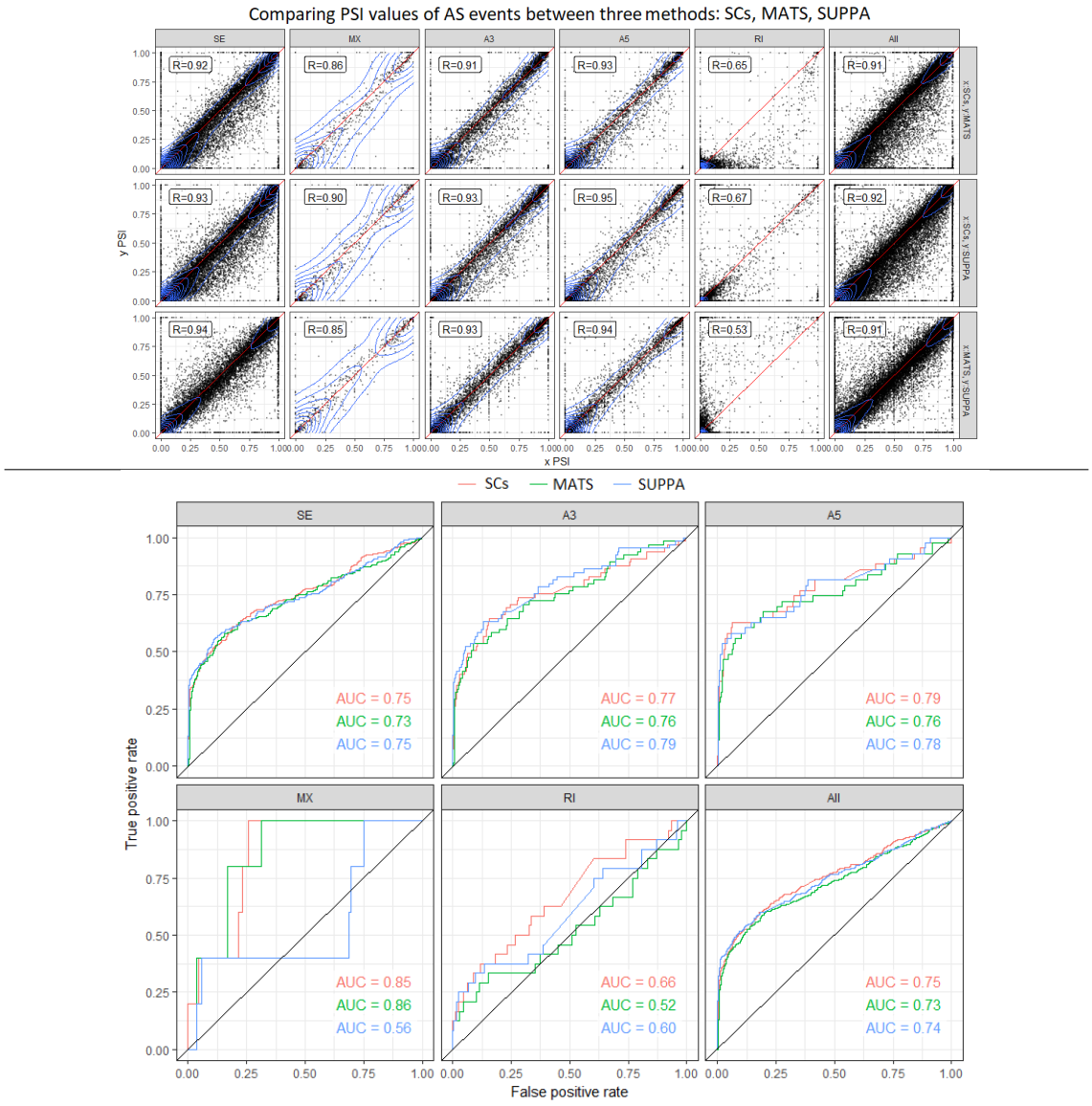

Figure 4.7: (**Top**) Comparing PSI values calculated using segment counts versus rMATS (first row), segment counts vs SUPPA (second row) and rMATS versus SUPPA (third row) on human samples from SwitchTx simulated dataset. Columns indicate seven types of alternative splicing events. (**Bottom**) Comparing ROC curves for differential alternative splicing using segment counts, rMATS and SUPPA for simulation dataset of switched abundance. Plots are stratified by event types. See Table 4.1 for number of events of each AS event type shown.

come in the expense of storage by storing large intermediate data (BAM files). The major motivation of transcript-based approaches is to achieve fast and light-weight pipelines that is not that expensive in terms of time and memory. For instance, even when using STAR, which is one of the fastest genome mappers in the field, using pseudo-alignment tools can be several orders of magnitude faster (or efficient in terms of storage and memory). That is why our segments approach is unique in leveraging such light-weight tools that utilizes pseudo-alignment algorithms with the capability of obtaining local measurements.

Table 4.2: Running time per sample (either single or paired-end reads) required by three approaches: using Segment Counts (SCs), counting-based (rMATS), isoform-based (SUPPA). Elapsed time is measured in minutes per pipeline including alignment/mapping step and the generation of PSI values (running using 64 threads on Dual E5-2690 2.90GHz). Both SCs and SUPPA approaches use RapMap for alignment, rMATS uses STAR.

Elapsed Time (mins)	rMATS (STAR)	SCs (RapMap)	SUPPA (RapMap)
Single-End			
Alignment	48	12	12
AS Quant	< 1	< 1	< 1
Paired-End			
Alignment	99	30	12
AS Quant	< 1	< 1	< 1

Among the five different splicing types exon skipping, alternative 3' and alternative 5' events gives the highest correlation between segment counts and rMATS approaches. In our experiments we noticed that rMATS (v4.0.1) does not behave as intended for intron retention events. We noticed that counts including junction reads only and counts including both junction and intron reads (which we use in this study) are the same. In other words, rMATS fails to report reads spanning

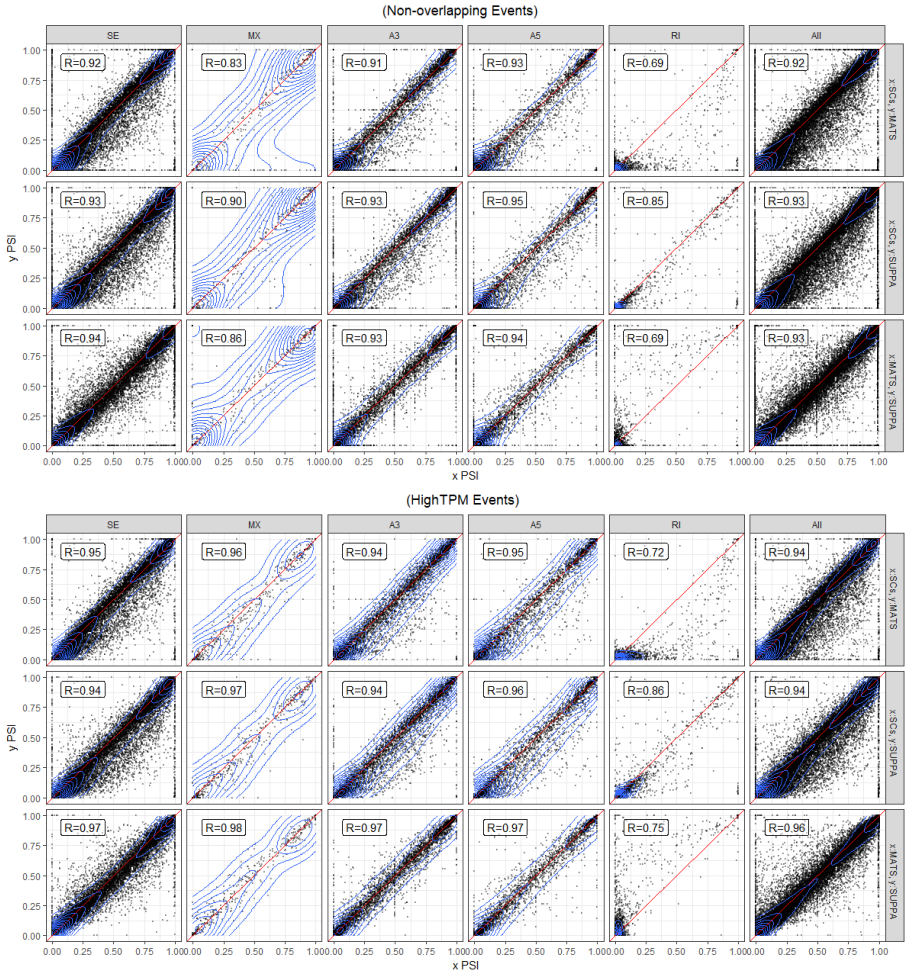

Figure 4.8: Comparing PSI values calculated using segment counts, rMATS (based on STAR's spliced alignment to genome) and SUPPA (based on estimated TPMs from kallisto's pseudo-alignment and quantification). Plots are stratified by event types. Plots are shown for two subsets of filtered events: non-overlapping events, events with high TPM in the annotation. See Table 4.1 for number of events of each AS event type shown.

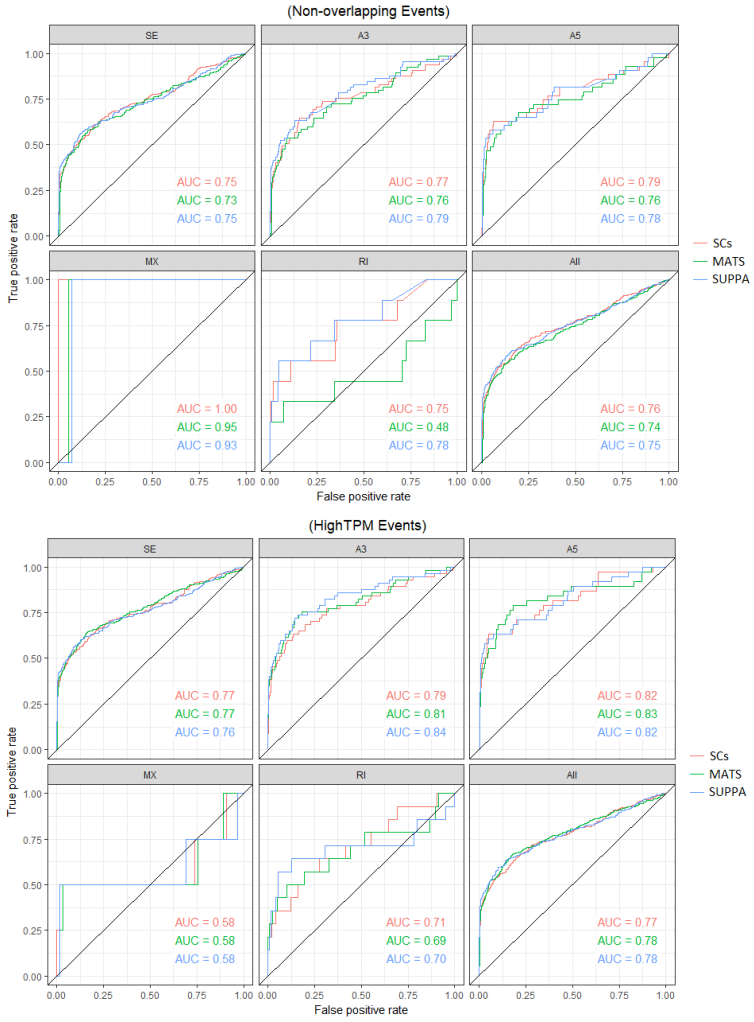

Figure 4.9: Comparing ROC curves for differential alternative splicing using segment counts, rMATS and SUPPA for simulation dataset of switched abundance. ROC curves are stratified by event types. See Table 4.1 for number of events of each AS event type shown. Plots are shown for two subsets of filtered events: non-overlapping events, events with high TPM in the annotation.

the intron, which explains the underestimated inclusion counts and PSI values for retained introns.

4.4 Segment-based Differential Alternative Splicing

Since the scope of this chapter is to introduce the use of segment counts as a statistic for studying alternative splicing, we want to use the simplest statistical model for differential splicing to exclude any advantage obtained by the model itself. In that matter we used the PSI values of the three approaches (SCs, rMATS, SUPPA) as discussed in the previous section. Then we used a linear model for differential hypothesis testing (implemented with Limma-voom R Package [43, 44]). However, more advanced models of differential analysis can be used instead. For example, a similar model to SUPPA2 can be developed to test the significance of ΔPSI by considering all events genome-wide [25]. Figure 4.7 (Bottom) shows ROC plots for sensitivity and specificity measures. Using segment counts achieves comparable performance to both rMATS and isoform-based approaches in that setting.

4.5 Simulation Datasets

Simulation of Switching Abundance (SwitchTx): We used the simulation data provided by [38] for both fruit fly and human organisms (E-MTAB-3766). Each dataset consists of six samples from two conditions. Each condition has three replicates. The reads for the replicates are simulated from real RNA-seq samples, to get realistic expression values, after incorporating a variance model and the change

required between conditions. The simulation is restricted to protein-coding genes in the primary genome assembly. The difference in transcript usage across conditions was simulated in 1000 genes randomly selected from genes with at least two transcripts and high enough expression levels. For each of these 1000 genes, the expression levels of the two most abundant transcripts is switched across conditions. Refer to [38] for full details of the preparation procedure of the dataset.

Simulation of Incomplete Annotation (IncompTx): Starting from the transcriptome annotation of the human genome, we searched for candidate cases where one combination of splicing events can be missing from the annotation. For a given gene, a combination of two splicing events (e_1, e_2) can form a candidate case if two conditions are satisfied. 1) If the two splicing events (ordered by their genomic coordinates) have at least one transcript common in their inclusion splicing $T_1^{inc} \cap T_2^{inc} = T_c^{inc}$ while there are no transcripts common between the inclusion of the first event and exclusion of the second event $T_1^{inc} \cap T_2^{alt} = \phi$ (which will later form the missing isoform in that gene). 2) If the transcript sets T_c^{inc} and T_2^{alt} share "long enough" contig in the splice graph between the two events. In our simulation, we searched genes on the forward strand for only combinations of SE, A3, A5 typed events. We used a cutoff of 100bp required for the common contig between the two events to be long enough. 2454 genes were found as candidate cases of possible missing isoforms and were used to simulate the data. In each of these genes a single novel isoform is formed by combining the inclusion splicing path of the first event with the alternative splicing path of the second event. Then we used polyester [39] to simulate RNA-seq reads (100bp single end reads) including the novel isoforms

which were given high expression levels.

Experiments run throughout the RNA-seq chapters used Ensembl GRCh37 and BDGP5 (unless mentioned otherwise) reference genomes and transcriptomes for human and fruit fly annotations, respectively.

4.6 Discussion

Recent work done on alternative splicing, e.g. Whippet [45] and ASGAL [46], may seem similar to Yanagi's approach since they all rely on processing the splice graph. ASGAL uses graph-based alignment approach to align reads directly into the splice graph which may introduce more complexity processing and traversing the graph. Whippet prepares and indexes what it defines as contiguous splice graph (CSG) before linear alignment of reads is performed. Both methods are built solely for the purpose of alternative splicing analysis. Yanagi's motivation and objective is different. It is important to note that the intent of this work is not to propose another alternative splicing method, but rather to introduce a conceptual framework that extends pseudo-alignment techniques through decoupling the alignment and quantification steps to generate statistics suitable to a variety of downstream analyses, including alternative splicing.

For the moment, analysts using ultra-fast pseudo-mapping will need to decide if they prefer possible loss of performance in AS analysis from using only local information, or from using an incomplete annotation. We believe that the results we show in this chapter are informative in this situation. In Section 2.6, we showed

how severely an incomplete annotation can decrease the correlation of PSI estimates with the truth (0.6 compared to 0.9 when using segments). Incomplete annotations are common in species with multiple introns per gene because the standard is to report a parsimonious set of transcripts rather than a complete set that represents all combinations of local splicing choices. We also showed in Section 2.8 an analysis on simulated data where the annotation is complete comparing the performance of the segments approach to an approach that makes use of information from other parts of the transcript (SUPPA). We observed that segment-based PSIs, that didn't use the information in the other parts of the transcript unlike transcript-based PSIs, obtain a 0.92 correlation with those PSI values estimated using that information. Given these results indicating there is greater loss of performance when using an incomplete annotation compared to the exclusive use of local information, we suggest that a conservative approach based on segment counts, which is more robust to incomplete annotation, is used for AS analysis.

Alternative Splicing (AS) methods that use transcript abundance, provided that a complete transcript annotation and a transcript quantification method that sufficiently addresses coverage bias across a transcript is used, can provide an advantage over methods that only use local information for AS analysis, including AS based on segment counts produced by Yanagi. Nonetheless, as we discussed elsewhere in the manuscript, there is no loss of information in segment counts and they may be used to perform transcript quantification or as statistics into a AS method that borrows information across splicing events to take advantage of their correlation.

This type of extension on the use of segment counts to perform transcript quantification is a fruitful direction for future research. Another interesting extension of our work would be to study the use of segments in discovering novel transcripts. Using paired-end reads mapped to two segments that do not share any common transcripts can be a potential direction.

Chapter 5: Segment-based Transcriptome Quantification

5.1 Overview

One main advantage of the Yanagi-based pipeline is that it decouples the transcriptome quantification task from the pseudo-alignment pipeline where it is actually unnecessary for performing analyses on both alternative splicing and gene level resolutions, leaving it only necessary when transcript-level analysis is targeted. In this section we discuss a quantification approach that uses the segment counts and their annotation structure to provide abundance estimates for the annotated isoforms in the reference transcriptome.

5.2 Problem Definition

Given a set of annotated transcripts T, the set of segments S created by Yanagi from that set of transcripts (i.e. $Txs(s_i) \in T; \forall s_i \in S$), and the segments-to-transcripts mapping maintained in the metadata of the segments which we will

represent by the indicator function

$$\forall s \in S \text{ and } t \in T : \gamma(s,t) = \begin{cases} 1 & \text{if } t \in Txs(s) \\ 0 & \text{otherwise} \end{cases}$$

In addition to the annotation information, the outcome of the alignment step for each sample obtained in the form of segment counts (or segment-pair counts in case of paired-end reads). We denote the set of segment counts as SC, where a count of single segment s_i is denoted c_i representing the number of fragments aligned to segment s_i. Equivalently for paired-end experiments, a count of segment-pair $< s_i, s_j >$ is denoted c_{ij} representing the number of fragments where one end aligned to s_i and the other end aligned to s_j. For the next few sections, we will describe the model assuming single-end reads for simplicity, but the ideas are extended for handling segment-pairs accordingly.

The objective of the quantification step is to correctly estimate the true relative abundance of each transcript present in a sample. One quantity measure of relative abundance is Transcripts per Million (TPM). For a specific transcript t_i, to have a relative abundance of TPM_i means that in a collection of one million transcripts, it is expected to have TPM_i number of copies of transcript t_i.

Similar to several state-of-the-art tools in the field [7–9], we adopt a generative model for the RNA-seq reads as being sequenced from a certain transcript proportional to the relative abundance of that transcript and its effective length. That is the general model assuming no sequencing bias. However, in the presence of bias,

the generative model is adapted to reflect the change in likelihood of generating fragments from each position within the transcript. For the next section, we focus on discussing the model on a bias-free assumption, leaving the bias correction into a separate discussion later.

5.3 Segment-based Expectation Maximization (EM):

Denote: c_s count for segment s, $\tilde{\ell}_s$ effective length of segment s, and $\tilde{\ell}_t$ effective length of transcript t. With a generative model under equal probability across transcript (no coverage bias), we define the probability that a read is generated by transcript t as $p(t) = \theta_t$. The conditional probability of transcript t to generate a read in segment s is denoted $p(t|s) \propto \frac{\theta_t}{\ell_t}$. The likelihood function of a set of RNA-seq fragments F can be defined using the following function $L(\theta) \propto \prod_{f \in F} \sum_{t \in T} \tilde{\gamma}(f,t) \frac{\theta_t}{\ell_t}$, where $\tilde{\gamma}(f,t)$ is a step function indicating whether fragment f is compatible with transcript t [7].

In order to optimize the likelihood function, we use EM algorithm to iteratively optimize the parameters θ in a factorized fashion where fragments are factorized into segment counts. The EM algorithm consists of two steps that are performed iteratively until every transcript has a change in the number of assigned reads $\Delta\theta_i N_g < 0.01$ from an iteration to iteration or the maximum number of iterations is reached.

E-Step: In this step, given the current estimates of θ_t, the algorithm computes

the conditional probability $p(t|s)$ as follows:

$$p(t|s) = \frac{\gamma(s,t)\frac{\theta_t}{\tilde{\ell}_t}}{\sum_{t' \in T} \gamma(s,t')\frac{\theta_{t'}}{\tilde{\ell}_{t'}}}$$

M-Step: In this step, given the current conditional probabilities, the algorithm adjusts the parameters θ_i to maximize the likelihood of the observed segment counts:

$$\theta_t = \frac{\sum_s p(t|s)c_s}{N_g}$$

5.3.1 Implementation Details in Yanagi:

Unlike the factorization approach in current methods [7, 8] to use equivalence class (EC) counts, the use of segment counts for factorized EM allows more genomic location separation in most cases. Since most segments are gene-specific, that divides the full-scale likelihood function that EM tries to optimize into a set of independent subproblems on the gene level which can be a desired feature to achieve better scalability through parallelism. The exceptions are cases where genes overlap or generally share common sequences due to gene duplication or some other biological causes. In such cases, the alignment steps will report some reads (hence read counts) as multi-mapped to multiple segments that could come from one or more genes. In such cases it is essential to run the EM for that cluster of genes together, in order to resolve the ambiguity in assigning those multi-mapped reads to their most likely gene and hence transcript.

Consequently, our implementation of transcriptome quantification first per-

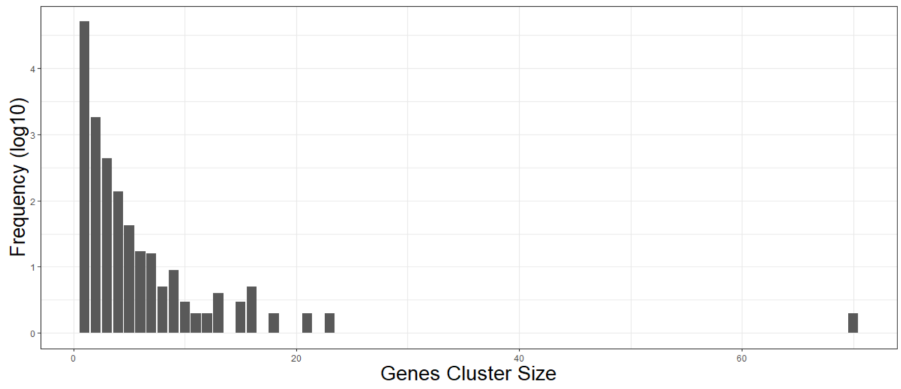

Figure 5.1: Histogram of the number of genes clustered together for individual EM runs. Out of 56,515 genes present in the human annotation (genome build Hg37), 50,219 clusters were formed. 95% of the clusters contain a single gene, whereas there is a single cluster with 70 genes.

forms a procedure while loading the annotation and segment counts information in order to form data-driven clusters of genes. A cluster of genes is formed if a significant read count is observed to be multi-mapped across segments from those genes (a count of three or more reads has to be observed for it to be considered significant). Once a cluster of genes is identified, a single call for the EM procedure is invoked with the combined input from each gene. e.g. for a cluster C of the set of genes $g_i, .., g_k$, the combined input includes $T_c = \cup_{g_i} T_i$, $S_c = \cup_{g_i} S_i$, $SC_c = \cup_{g_i} SC_i$ and $\gamma_c(s,t) = \gamma^{g_i}(s,t)$ if $s,t \in g_i$. Figure 5.1 shows the distribution of gene cluster sizes formed from human genome (annotation Hg37) driven by simulation data described in section

5.4 Preliminary Results (Complete Annotation Case):

In this section, we evaluate the performance of our segment-based quantification against one of the state-of-the-art k-mer based methods Kallisto [7] on simu-

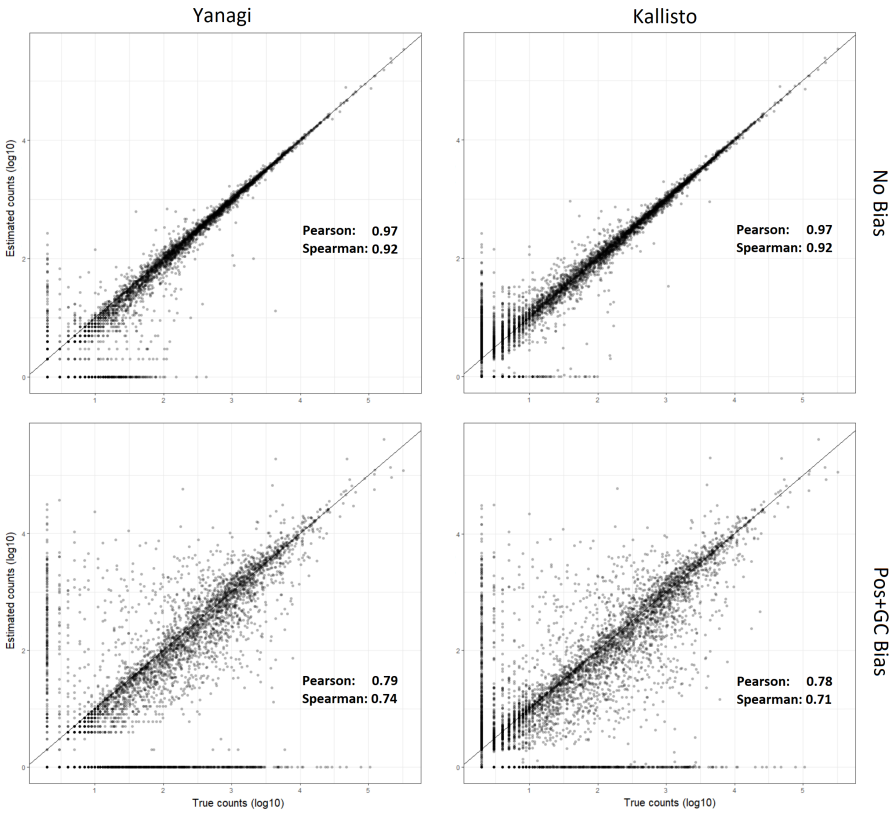

Figure 5.2: Plotting estimated raw counts for each transcript (using Yanagi and Kalisto) against the true counts in two simulated datasets, one with no coverage bias and another with both positional and GC biases. Yanagi and Kallisto show comparable performances.

lation data. We used Polyester [39] as a simulator to generate two datasets. One without any biases, and another with two common sources of biases: positional bias and GC bias [17].

Both current implementations of Yanagi and Kallisto do not have elaborate model handling for such sources of biases, resulting into some drop in performance in case of coverage bias. However, we believe our segment-based approach provide a natural framework to account for these sources of bias since the count statistics are based on location-preserving factors. Yet we leave tackling that challenge for future extensions.

5.5 Quantification with Incomplete Annotation:

In chapter 4 we started the discussion on the possibility of having incomplete transcriptome annotation and its negative effect on transcript-based approaches for alternative splicing analysis and showed an example of *Drosophila melanogaster* for a known complex and incomplete annotation. Here we directly tackle that problem since we are discussing transcript-level analysis.

Intuitively, the problem of incomplete annotation is plausible when dealing with RNA-seq. One possible cause are issues arising from a transcriptome assembly stemming from an its accuracy, comprehensiveness and parsimony. Annotation completeness would also depend on the diversity of transcriptome data across tissues used to construct it. These issues are more prevalent in complex and poorly studied organisms. Another type of plausible causes could arise in the presence of novel splicing events present in specific biological conditions of interest. Consider some rare disease where cells undergo some novel splicing that is not present in the reference annotation. Without a framework handles such possibilities, these cases would go unidentified and may lead to wrong or biased conclusions. Hence, it is essential to tackle those challenges.

5.5.1 Possible Missing Junctions:

A transcript is defined as a uni-directed sequence of splicing events of expressed genomic regions (exons or retained introns) along the genomic axis. Therefore to break down the problem of unannotated transcripts, it can be viewed as either *(1)*

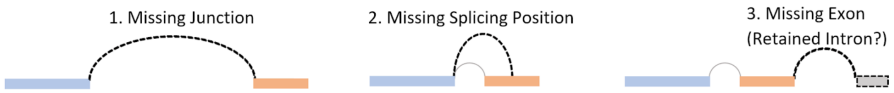

Figure 5.3: Diagram of possible scenarios of missing junctions from annotation. **Type I.** only the junction splicing is missing. **Type II.** missing splicing position. **Type III.** missing exon splicing (both the junction and the exon are missing)

missing one of more splicing events from the annotation, *(2)* having an unannotated combination of individual annotated events, or *(3)* a combination of both cases together. Figure 5.3 illustrates three possible cases of missing junctions from the annotation. For the rest of this Chapter, we are focusing only on missing junction type (1) where only the junction itself is unannotated as a proof of concept, leaving the other more complex cases for future extensions of this line of research.

When considering the missing junction type (1) from the perspective of the annotated transcripts, it could be viewed as one of two possibilities: either the pair of exons on the two ends of the junction are never co-spliced (there is not a single annotated transcript that splices-in both exons), or there is at least one such transcript while the missing junction forms an immediate connection between the two exons. The second case could be much more challenging to detect depending on the difference in path distance between the annotated path and the missing path. In other words, it would be much harder to detect if the two exons are a few bases apart within some annotated transcript.

Our segment-based approach for novel junction discovery targets both cases using paired-end reads. During the alignment step, fragments that are reported as unaligned are further examined for the possibility of spanning a novel junction. If both read ends are aligned to two different segments with high scores [47], whilst

there was no common annotated transcript between the two segments, then that read will be counted as an evidence of the first case. Whereas if a fragment was reported as unmapped while both ends have alignments of high score yet it was rejected by the aligner because that mapping doesn't conform with the expected fragment length distribution observed from the data, then it would be counted as an evidence of the second case. Alignment step in Yanagi outputs a separate file containing such counts in the form of segment-pair counts as counts for novel junctions which are used as input in the quantification step.

5.5.2 Incomplete Annotation Bias:

To evaluate the bias effect of incomplete annotation on the performance of state-of-the-art quantification methods, we performed a simulation procedure to produce scenarios of missing junction type (1). We adopted that procedure on human transcriptome Hg19 for experimental results on this chapter, however, the same procedure could be extended to other genomes as well.

Simulation Setting: First task is to prepare a version of the annotation which is incomplete by defining a set of transcripts to be dropped from the original annotation. That set of transcripts will represent the missing transcripts from the annotation during the alignment. For a transcript to be considered as a candidate dropped transcript, it must have a *unique* junction of type (1). The segment formulation facilitates identifying that uniqueness condition as follows:

$$\exists s_i \text{ s.t. } len(s_i) = 2(L-1) \text{ and } |Txs(si)| = 1 \text{ and } \nexists s_j \text{ where } Txs(s_i) = Txs(s_j)$$

Given the transcripts abundance profile of one sample from the dataset used in chapter 3, we randomly selected transcripts to be dropped proportional to their relative abundance (TPM) such that only one transcript is dropped from any given gene and with at least $TPM > 10$. That criteria resulted on 944 genes with one transcript to be dropped from each gene. The incomplete annotation is formed by removing those selected transcripts from the full set of transcripts. Next step is to simulate paired end reads from the full set of transcripts (i.e. using the complete annotation) using polyester [39]. We generate two datasets, one with no bias and another with positional and GC biases. That generates samples of around 38 million pair-end reads per sample. Lastly, Yanagi is used to generate segments from the incomplete annotation. That segments library will be used as the reference during the alignment and quantification steps.

Quantification Performance

First, we want to stratify the resulting performance depending on the abundance of the missing transcript. Based on the way the simulation setting is prepared, we obtain the distribution shown in figure 5.4 of the relative abundances of the missing transcript (measured on TPMs) from the 944 genes.

Figure 5.5 shows a comparison between the performance of the quantification process under complete or incomplete annotation on the same dataset defined earlier in this section. We prefer to use the local transcript fraction θ_t [48] as our metric here since it is normalized between 0 to 1 in order to show the bias impact on all genes regardless of the total gene expression level. For transcript t in gene g the local transcript fraction is calculated as $\theta_t = \frac{count(t)}{count(g)}$. The plot is stratified (columns)

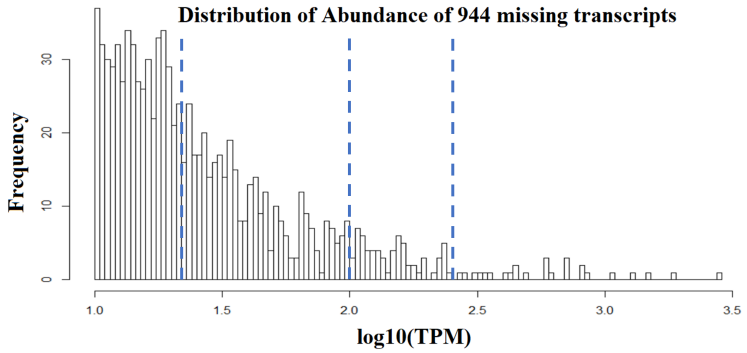

Figure 5.4: Histogram of the relative abundance of 944 transcripts removed from the annotation to simulate genes with a single missing transcript. Three dotted vertical lines show the stratification thresholds we use to distinguish performance on four groups of genes.

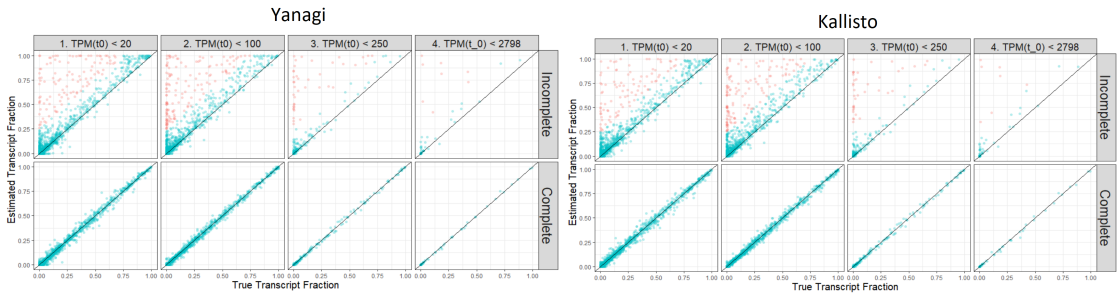

Figure 5.5: Performance of quantification when complete and incomplete annotation is used on the same dataset for Yanagi **(left)** and Kallisto **(right)**. Points represent local transcript fractions within a gene. Each plot is stratified (columns) into four groups according to the thresholds shown in figure 5.4. Red points represent transcripts with absolute difference between true and estimated fractions > 0.25. Pearson Correlations of Yanagi are 0.99 and 0.86 for complete and incomplete annotation respectively. Pearson Correlations of Kallisto are 0.99 and 0.84 for complete and incomplete annotation respectively.

into four groups according to the thresholds shown in figure 5.4. Performance clearly suffers under the incomplete annotation setting. Both methods tend to overestimate the abundance of many transcripts with more observed overestimations when the missing transcript have a moderate to high abundance (e.g. more red points in the second group than the first group even though the first group have more overall points).

These preliminary results show how bad the performance of standard quantification approaches can get under the presence of unannotated transcripts in RNA-seq data and that motivates our attempt to tackle that problem in the next few sections to correct for incomplete annotation bias in transcript abundance estimation.

5.6 Incomplete Annotation Bias Correction:

The problem of removing the negative impact of quantifying over an incomplete annotation may sound difficult to be solved without attempting to assemble the missing transcripts at first; since uniquely mapped reads to the missing transcripts are likely to end up not being mapped while most ambiguous reads from those transcripts will be aligned to other annotated transcripts ending up overestimating their abundances to some extent because algorithms like EM prioritize assigning all mapped reads to one annotated transcript. However, we take a different approach to tackle that problem by trying to reduce that bias in the annotated transcripts without attempting to assemble the missing set of transcripts. Yet another use of segment-level counts from Yanagi is apparent in that challenge, showing the power of our segment statistics and framework.

Since segments carry along their positional information within each transcript that each segment is compatible with, that could provide a possible direct measure of the observed read coverage within each transcript. We utilize that observation to identify transcripts that have large differences between observed and expected coverage across one or multiple regions of the transcript. In other words, segments

0. Run EM on the annotated transcripts T, given: S, SCs, $\gamma(s,t)$. Outputs: $\{\bar{\theta}_t\}$
1. Determine genes $\{g_{inc}\}$ with significant new Segment Pairs $\{sp_0\}$
   (assuming these junctions fairly reflect real abundance of the missing transcripts $\{t_0\}$)
2. For each gene g_{inc} and for each segment in it, calculate **Expected** vs. **Observed** coverage:
   2.1. $\mathrm{E}(s_i) = \sum_{t \in s_i} \frac{c(t)}{\tilde{l}(t)}$
   2.2. $\mathrm{O}(s_i)$ is observed segment counts, normalized by $\tilde{l}(s_i)$
   2.3. $\Delta cov(s_i) = \max[\mathrm{E}(s_i) - \mathrm{O}(s_i),\ 0]$, $\delta(s_i) = \begin{cases} 0\ ; \Delta cov(s_i) = 0 \\ 1\ ; otherwise \end{cases}$
3. Assume $\bar{\theta}_t = \theta_t^{true} + \partial_t\ \forall t \in T$: estimate ∂_t
   Optimize how these errors should be distributed among the annotated transcripts,
   constrained by the value of sp_0

$$\min_{\partial} \left[\sum_{S} \left\| \Delta cov(s_i) - \delta(s_i) \sum_{t \in s_i} \partial_t \right\|^2 + \lambda \left\| cov(sp_0) - \sum_{T} \partial_t \right\|^2 \right]; s.t.\, \partial_t \in [0, \bar{\theta}_t]$$

Figure 5.6: Procedure for incomplete annotation bias correction based on segment counts.

represent such regions and we calculate the expected and observed coverage and hence the residuals in segment coverage $\Delta cov(s_i)$. The intuition is to calculate the error in the estimated abundances of each transcript then adjust those abundance estimates to minimize an error function guided by the calculated residuals in segment coverages and the novel segment-pair counts that provide evidence for missing junctions (as discussed in Section 5.5.1).

Our bias correction procedure begins after the EM algorithm is run and for genes with significant counts observed for a new segment-pair. That measure of significance can be a user-defined parameter, or it can be calculated as a percentage of the overall gene expression observed in the sample. Each of those genes is identified as a candidate gene with possible incomplete annotation and the annotation bias correction procedure is run. Figure 5.6 shows the steps of that correction procedure. We define the abundance residuals for each transcript as $\delta_t = \hat{theta}_t - \theta^t rue_t$ where $\hat{theta}_t$ is the estimated abundance from running EM. These residuals are used to

minimize the error function shown in figure 5.6. That function consists of two error terms. The first term entails the abundance residuals to explain the difference in coverage of the corresponding segments. The second term on the other hand restricts the sum of the abundance residuals to match the coverage of the novel junction discovered through the new segment-pair.

5.7 Preliminary Results (Incomplete Annotation Case):

As a preliminary experiment, we evaluate our correction approach over the simulation data prepared as explained in section 5.5.2. Since the optimization function depends on the new segment-pair counts that are obtained from the alignment step, we wanted to evaluate the strength of our correction model independently from that junction discovery step. Therefore, we ran the correction procedure assuming we have a perfect observation of that novel junction. We calculate sp_0 from the true abundance of the missing transcript. That would work as a practical upper-bound of how well our correction model can get. Figure 5.7 shows the estimated abundances (as transcript fractions) against the truth and compares that with the estimates before correction. Our correction procedure was able to recover most overestimated abundances leading to a Pearson correlation of 0.97 instead of 0.86 before correction.

Another essential aspect to evaluate is the dependency of the correction model on the new segment-pair counts; how sensitive our model to that discovery step. Thus, we calculated various levels of sp_0 reflecting different observation efficiencies during the junction discovery step. We calculated sp_0 based on 100%, 70%, 50%,

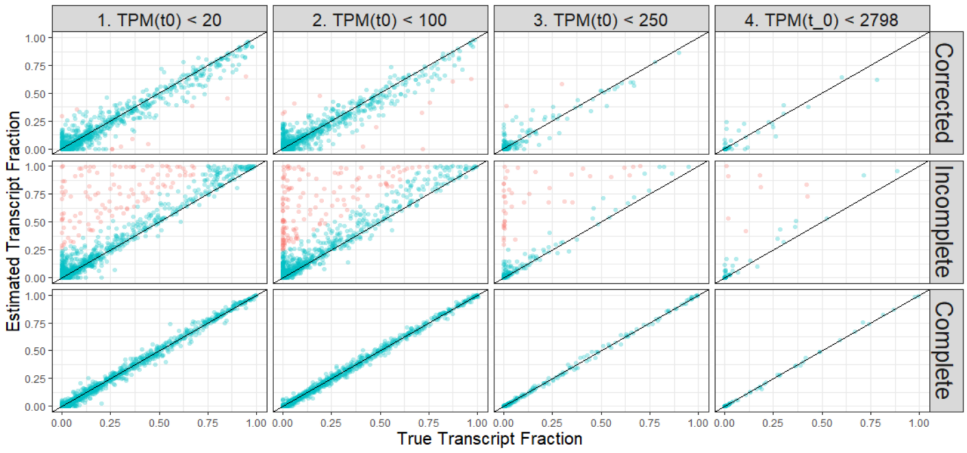

Figure 5.7: Performance of the correction procedure on simulation data (Corrected). It shows the estimated abundances (as transcript fractions) against the truth and compares that with the estimates before correction (Incomplete) and when the complete annotation is used (Complete). The correction procedure was able to recover most overestimated abundances leading to a Pearson correlation of 0.97 instead of 0.86 before correction.

30% of the true abundance of the missing transcript and ran our correction model under each setting. Table 5.1 shows the Pearson correlation under each scenario. The table shows a small drop in performance going from 100% to 70% or even 50% which is reassuring that our model is not highly sensitive to sp_0 and it potentially could have positive effect in reducing the annotation bias even if the observed coverage of the novel junction was relatively far from its true coverage.

Table 5.1: Pearson correlation of the true and estimated abundances (transcript fractions) after correction for incomplete annotation, under different levels of calculated sp_0. Each level represents if the new segment-pair is observed with a count reflecting k% of the true abundance of the missing transcript. The correlation is shown for 100%, 70%, 50%, 30% and no correction at all. Figure 5.8 shows the scatter plots of estimated abundances for both 70% and 30% runs.

	100%	70%	50%	30%	No Correction
Pearson	0.97	0.96	0.95	0.93	0.86

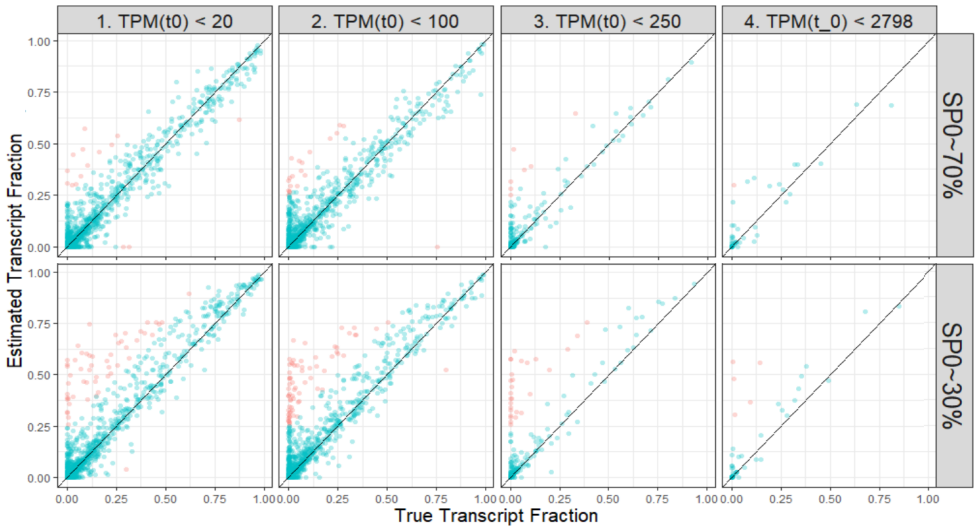

Figure 5.8: Performance of the correction procedure on simulation data given an observed coverage of novel junction calculated from 70% (Top) and 30% (Bottom) of the true abundance of the missing transcript.

5.8 Discussion

In this chapter we have shown how our Yanagi framework can be used to perform transcriptome quantification. We have shown that using segment counts as summarizing statistics of the RNA-seq sample can give comparable results with k-mer based pseudo-alignment approaches in the regular cases of quantifying over a complete annotation. Moreover, we tackled a more challenging case where transcriptome quantification is performed in the presence of unannotated transcripts. We have shown through simulations how algorithms like EM can overestimate abundances of the annotated transcripts. Then we proposed a segment-based bias correction procedure that showed positive results an effectiveness on preliminary experiments. Here we demonstrated a proof of concept of how the bias caused by the incomplete annotation can be corrected without the need to perform assembly of

the full splicing path of the missing transcripts. In order to establish more solid conclusions, the experiments present here must be extended on real data, comparing the performance with other assembly-based approaches and handle the other possible cases of missing junctions.

Chapter 6: Bridging Linear to Graph Alignment for Whole Genome Population Reference

6.1 Overview

As mentioned in Chapter 2, the linear approach for building a population genome reference is not preferred mainly for two reasons. First, it is very inefficient representation of sequences that inherently share a majority of their subsequences, especially with a large number of sequences. Consequently, the presence of multi-mapped reads becomes more likely and adds ambiguity during quantification. Second, it does not provide fine-grained information regarding the structure of the alternative alleles which is essential when studying specific variants. The same two challenges are present in RNA-seq.

In RNA-seq, the input graph that Yanagi processes is a splice graph where vertexes mainly represent exons (expressed regions in the genome axis) and paths represent transcripts. Whereas the sequences subject to Yanagi's segmentation in our case here represent alternative alleles instead of transcripts. Thus, the input graph in our case would be both location and sequence dependent rather than being solely location dependent. Fortunately, Yanagi's method is agnostic of what the

input graph actually represents. In other words, the same segmentation approach can be adopted in problems where representing data as graph is desired yet processing the graphs can be challenging. In that sense linearizing the graph can assist in balancing the efficient structural representation of the data while maintaining processing simplicity, regardless of what the vertexes and paths represent.

6.2 Population Alleles MSA Graph

We begin by projecting the population genome graph into a similar format that Yanagi originally accepts as an input. First, we prepare the multiple sequence alignment of the provided alleles. Some databases already provide the alleles' MSA (e.g. IPD-IMGT/HLA Database for HLA genes). If not available, MSA can be derived given the list of variants (from VCF file) or by running some multiple sequence alignment algorithm offline over the list of available alleles. In the scope of this chapter, we assume that the MSA data is already provided as an input to our method. Figure 6.1 shows how our method takes the alleles MSA as an input to construct a graph format similar to that which Yanagi accepts.

The first step is to build the equivalent graph representation (MSA Graph) of the allelic sequences obtained from the MSA step (Figure 6.1 A). The MSA Graph is constructed as a partial-order graph [49]. The MSA Graph is a sequence of layers (dashed rectangles in figure 6.1 B) where a layer contains one or more vertices reflecting the alternative sequences in the set of bases corresponding to that layer. In other words, edges in the graph are formed exclusively between vertexes in

adjacent layers, while no edges can exist between vertexes of the same layer. Note that the graph is collapsed during construction to form a compact representation with the least number of vertexes and edges. Thus, the width of a vertex (the number of bases it represents) varies depending on the polymorphism of the region. Each vertex is also labeled by the set of alleles containing this vertex's sequence.

The projection of MSA graph into Yanagi's standard graph is achieved by applying two modifications into the MSA graph. First, we flatten the graph. By flattening we mean eliminating the layered structure maintained during constructing the graph. That can be obtained by sorting vertexes according to a breadth-first search (BFS) traversal mechanism. The second modification is handling INDELs. Recall that an INDEL is represented in the MSA sequences as a sequence of dots in either reference allele (insertion) or in an alternative allele (deletion). INDEL vertexes are eliminated from the flattened MSA graph and replaced by edges connecting the preceding and the following vertexes of the INDEL vertex. Note that, unlike the original MSA graph, the flatten MSA graph may contain edges connecting vertexes far apart, depending on the INDEL's length and the polymorphism of the region (Figure 6.1 C). Moreover, alleles labels are maintained by the edges in the flattened MSA graph rather than vertexes in the MSA graph.

6.3 Segment-based Linear Population Genome Reference

For each gene, a flattened MSA graph is built and processed separately. Then Yanagi is run to create a set of maximal L-disjoint segments as discussed in section

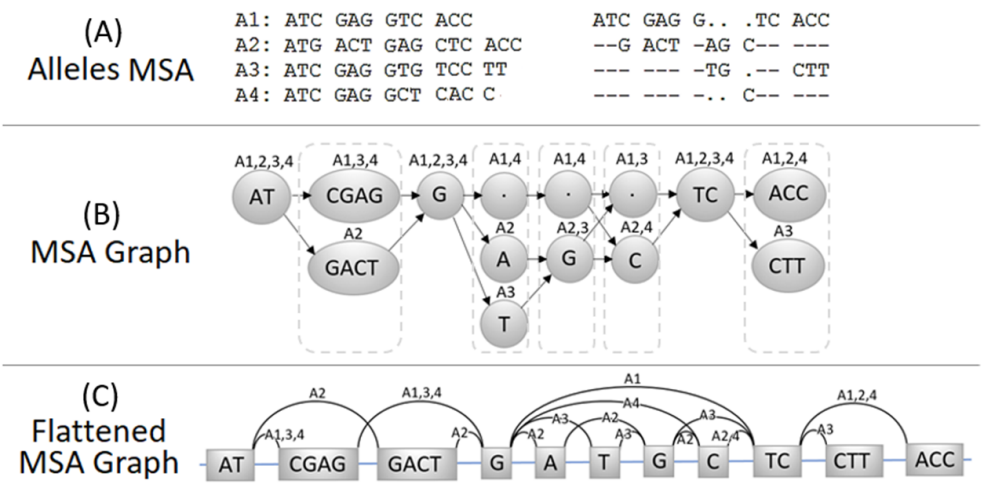

Figure 6.1: The process of preparing the population graph into the graph format of Yanagi. **(A)** Starts by preparing the alleles MSA. **(B)** Builds MSA Graph of each genes' alleles as a partial-order graph. **(C)** Flattening MSA Graphs and replacing INDEL vertexes by new edges.

2.3. Segment sequences are reported into FASTA file along with some header information regarding how the segment was formed, e.g. genomic coordinates and set of alleles it belongs to, in addition to a segmentID reflecting its corresponding geneID. The segments of each gene are then combined to form genes-specific or genome-wide population reference. If segments were created for only a specific set of genes (e.g. HLA genes), a genome-wide population reference is formed by combining the segments of these genes and the reference sequence of the rest of the genome (excluding the regions of these genes). Figure 6.2 shows how a segment-based population genome reference is formed by including allelic segments of two genes.

The resulting FASTA file can be considered an efficient linear reference of the population genome which preserves the relative structure of different alleles and maintains a sufficient level of disjointness between sequences. The segment-based genome reference can be then used by any alt-aware alignment algorithm. After

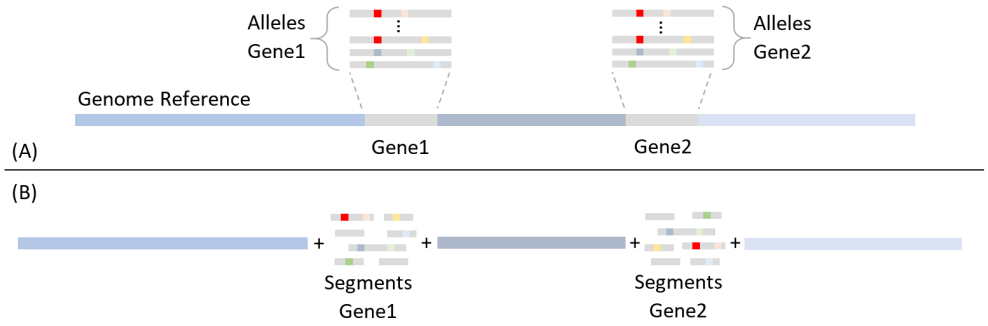

Figure 6.2: How a segment-based population genome reference is formed by including allelic segments of two genes with the rest of the genome reference. **(A)** shows the genome reference and the alternative alleles of two genes. **(B)** shows the resulting segment-based population genome reference produced by Yanagi.

reads are aligned, the resulting BAM file is processed to extract reads mapped into segments of the genes of interest. Extracting reads mapped to a specific gene can become as easy as scanning the BAM file for alignment records with the corresponding geneID.

6.4 HLA Segments Analysis

Our experiments in this chapter focuses on building segments-based linear population genome for HLA genes. However, the same approach can be applied to other genes as well. We obtained the set of alleles present from IPD-IMGT/HLA Database (as of May 2018). For each HLA gene, its alleles MSA is used to build maximal L-disjoint segments. In addition, we use the primary assembly in GRCH38 as our reference genome throughout the experiments.

Figure 6.3 shows the number of generated segments per gene (left) and the distribution of the sequences length (right) for $L = 150$. One may have concern that linearizing the graph may cause exponential growth in the number of segments

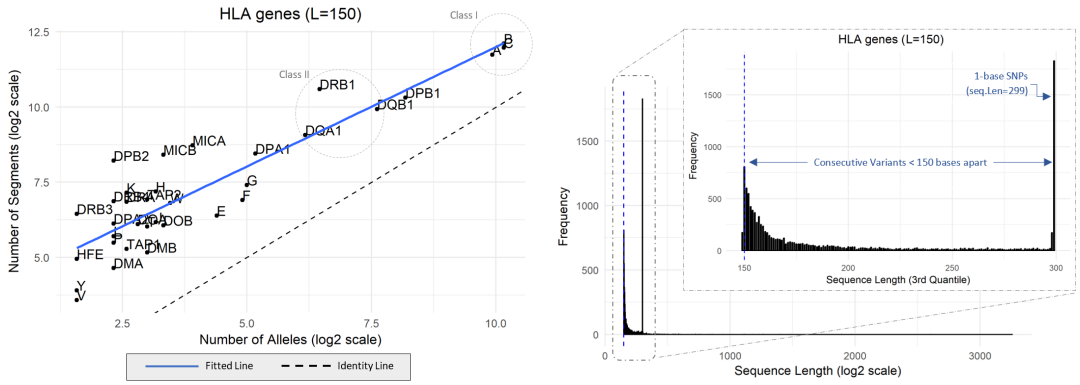

Figure 6.3: **Some characteristics of the generated segments (L=150) for HLA genes (based on alleles from IPD-IMGT/HLA Database, May 2018). (Left)** The number of generated segments versus the number of alleles per gene. The dotted line represents the identity line. **(Right)** Sequence length distribution of the generated segments with a zoom over the third quartile of the skewed distribution. Blue dotted line marks length=150. The distribution is bimodal with a peak at length of 299 which corresponds to 1-base SNPs. 1-base SNPs generates a segment spanning the SNP besides L-1 bases before and after the SNP. Segments falling into the third quartile range mostly correspond to cases where consecutive variations (including multi-base SNPs) are less than L bases apart. The distribution tail beyond 300 mostly represent long genomic regions absent from variations.

generated. However, the left panel shows that the number of generated segments grows linearly with the number of alleles per gene. On the other hand, the right panel shows the length distribution of the segments. 75% of the segments are of length within the range $[L, 2L]$. There is a peak at length of 299 $(2L-1)$ which corresponds to 1-base SNPs. 1-base SNPs generates a segment spanning the SNP besides $L-1$ bases before and after the SNP. In addition, most segments falling into the third quartile range correspond to cases where consecutive variations (including multi-base SNPs) are less than L bases apart. Whilst the distribution tail beyond $2L$ mostly represent long genomic regions absent from variations.

To study the final linear population genome generated from alleles of the six HLA genes, Table 6.1 and Table 6.2 summarizes two factors: k-mers found and

Table 6.1: Genome library size for the six HLA genes using reference+alleles concatenated and reference+segments ($L = 150$) in three metrics (reference-only and graph-based library sizes are provided as a reference when applicable). In case of graph, number of bases is estimated as the summation of bases of the graph nodes.

	Reference	Ref+Alleles	Ref+Segments	Graph
Number of bases (Gb)	0.045	9.25	2.39	0.048
Number of sequences	6	2,094	45,609	2,094
FASTA file size (MB)	0.03	10	2.4	NA

library size. In table 6.1, we compare the library sizes of the six HLA genes between using reference+alleles concatenated, and reference+segments ($L = 150$) in three metrics (reference-only and graph-based library sizes are provided as a reference when applicable). The total number of bases with segments is 4x lower than linearly concatenating alleles to the reference. Despite the increase in number of sequences when using segments, these sequences are much shorter which reflects on a smaller size FASTA file in case of segments.

Since we also target studying the use of segments with k-mer based lightweight alignment approaches, it was important to study the number of k-mers found in segments that are absent from the reference. Table 6.2 provides a detailed summary for each of the six HLA genes. Even with the low value of $k = 16$, around 36% of the k-mers found in alleles are novel to the reference. Consequently, it is expected for k-mer based approaches to highly suffer if used to align reads to only the reference. Whilst segments can empower such approaches with the new k-mers to achieve better alignment accuracy. That is part of our next analysis.

Table 6.2: Number of K-mers found in alleles of the six HLA genes ($K = 16, 30, 60$) and the percentage of new k-mers not found in the k-mers table built from the reference alone.

	(ClassI)			(ClassII)			
	HLA-A	HLA-B	HLA-C	HLA-DQA1	HLA-DQB1	HLA-DRB1	Total
Number of Alleles	976	1,144	1,141	72	196	88	3,617
Segments 16-mers	19,115	18,865	20,691	19,274	26,830	53,076	148,764
New 16-mers (%)	45.7	49.8	49.6	19.2	40.7	27.9	36.6
Segments 30-mers	34,088	33,198	36,124	27,795	39,031	75,509	237,349
New 30-mers (%)	85.4	97.1	96.9	70.5	99.1	88.5	90.4
Segments 60-mers	68,483	67,837	72,026	39,699	56,232	105,267	403,371
New 60-mers (%)	93.7	99.7	99.6	82.8	99.8	95.6	96.1

6.5 Simulated Datasets

We prepared two simulated datasets to perform different perspectives of the analysis. In both datasets, a sample is prepared by simulating reads from two alleles for each gene selected from the HLA alleles database. Paired end Reads of length 100 are simulated using *wgsim* tool [50] with read coverage 40x.

The first simulated dataset (SimData1) simply contains just three samples. Each sample represent different scenario. In the first sample, *ClassI-Easy*, reads are simulated from only ClassI genes (-A, -B, -C) and alleles are manually selected not much different from the reference. This would represent cases where a sample is very close to the standard human genome. While in the second, *ClassI-Hard*, alleles are selected to be relatively different from the reference. This sample would represent divergent samples from the reference genome. The third sample, *ClassII-Hard*, focuses only on ClassII genes (-DQA1, -DQB1, -DRB1). ClassII genes are much more polymorphic than ClassI, so alleles in ClassII are more diverse and

Table 6.3: Number of correctly aligned reads from simulated reads using: HISAT-genotype (graph aligner), BWA-MEM (linear alt-aware aligner), and RapMap (RNA-seq lightweight aligner). In case of both BWA-MEM and RapMap, results are shown either when using only the reference genome or the reference combined with Yanagi's segments for the six HLA genes.

	Num.	HISAT-	BWA-MEM		RapMap	
	Reads	genotype	Ref	Ref+Segs	Ref	Ref+Segs
ClassI-Easy	6,000	5,900	6,000	6,000	4,163	5,990
ClassI-Hard	6,000	5,966	5,797	6,000	3,553	5,990
ClassII-Hard	14,000	13,844	12,232	13,997	7,628	13,975

richer of variations.

The second simulated dataset (SimData2) aims at testing stability of results when a sample read of the whole genome is tested. This dataset contains 10 simulated samples. For each sample, two alleles were randomly selected from each of the six genes (ClassI and ClassII), then reads simulated from the six genes are combined with 2 million reads simulated from the rest of the genome.

Last experiment is done to test performance on real data by aligning reads (2x151bp) from replicate of NA12878 obtained from Illumina's NovaSeq: TruSeq PCR-Free 350 library.

6.6 Extracting reads from Simulated Data

Here we target the problem of reads extraction for the six HLA genes and the goal is to achieve high accuracy retaining reads sequenced from these genes. The problem of read extraction is crucial to many further analyses including HLA typing, since the more correct reads extracted for the genes, the more accurate the prediction of the typing model.

Table 6.4: Running time for alignment of sample NA12878 (24 threads on Dual E5-2690 2.90GHz)

	HISAT-genotype (Graph)	BWA-MEM (Ref+Segs)	RapMap (Ref+Segs)
Running Time	20 hours	8 hours	2 hours

Table 6.3 shows the preliminary results of running different approaches on simulated data (SimData1). The use of segments with linear aligners can boost the read extraction efficiency. By examining the separate scenarios of each sample, although the increase in the number of extracted reads may look insignificant for the first sample (ClassI-Easy), since the reads were simulated from allele that is not much different from the reference. Whereas the significance increases with the second sample (ClassI-Hard) especially when using RapMap. While reference only approaches misses a greater number of reads with the third sample (ClassII-Hard) where alleles are from ClassII genes which are harder and denser in variants.

We also experimented on the second dataset (SimData2) to examine the stability of the different approaches with several different alleles randomly selected. Figure 6.4 shows that aiding the two linear aligners with segments boot the method recall (ratio of correctly extracted HLA reads) back to(or even slightly higher than) the level of the graph aligner. Table 6.4 shows the difference in alignment running time between linear aligners with segments versus graph aligner (HISAT-genotype) that can reach up to 10x speedup.

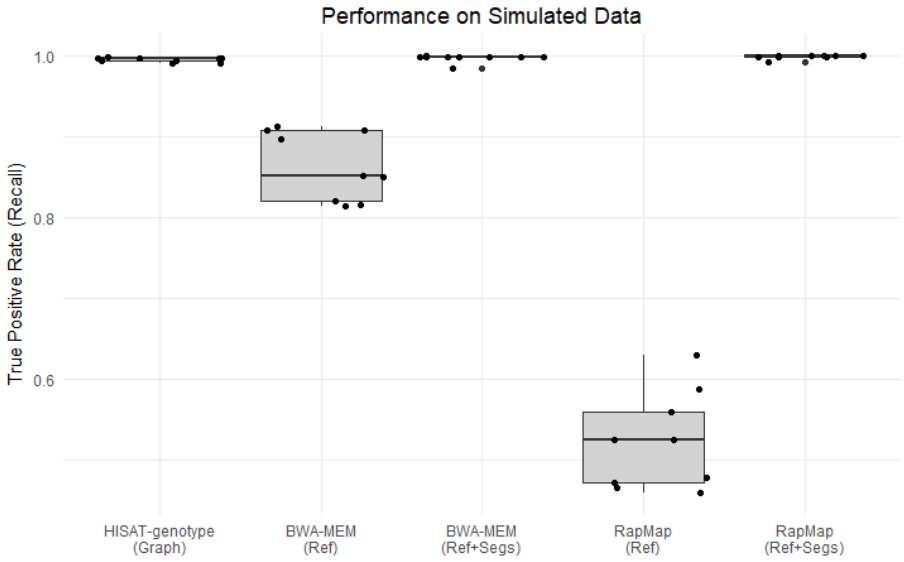

Figure 6.4: Simulation Dataset of 10 samples. Each sample of 56k HLA reads simulated from two randomly selected alleles for each of the six HLA genes. Plot shows recall rates of extracted HLA reads using 5 approaches: HISAT-genotype (graph aligner), BWA-MEM (linear aligner) with HG38 reference only, BWA-MEM with reference + HLA segments, RapMap (k-mer based lightweight aligner) with reference only, RapMap with reference + HLA segments. Both linear aligners performance are elevated compared to graph aligner when HLA segments are used.

Chapter 7: scGAIN: Single Cell RNA-seq Data Imputation using Generative Adversarial Networks

7.1 Overview

Until the recent advent of single cell methods, measurements of gene expression based on RNA sequencing (RNA-seq) were obtained in bulk over a cell population. Unfortunately, cell-to-cell variability in gene expression within the cell population is lost when RNA sequencing is applied in bulk. Single cell RNA sequencing (scRNA-seq) is designed to capture heterogeneity across cells within a cell population.

While providing a rich view into the heterogeneity underlying a cell population, data generated by current scRNA-seq technologies are much noisier and sparser than data obtained by bulk RNA-seq. Since the amount of mRNA captured from individual cells is usually low, single-cell assays may fail to capture molecules from transcripts with low-to-moderate expression in a considerable number of cells. Genes suffering from this phenomenon are referred to as dropdown genes. Another observed phenomenon is dropout (although recently shown to be less prevalent as originally thought in the literature [51, 52], and see Appendix B) where transcripts fail to be captured even in genes with high expression due to technical reasons. Furthermore,

because single cell datasets usually include a heterogeneous population of cells, the gene expression distribution in a single gene may show a large number of zeros as some cell types express the gene and other cell types may not. In other words, a gene expression can be observed as zero in a cell either due to biological (cell types where gene is not expressed) or technical (low sampling or capture failure) reasons. Typical downstream analysis of scRNA-seq data must deal with a highly sparse expression matrix of about 80-90% zeros, arising due to a variety of factors, which can bias the outcome of the analysis if dropouts are not carefully handled.

7.2 Related Work

Most clustering, cell type identification, and dimensionality reduction pipelines incorporate methods to handle zeros in their models either through implicit imputation (e.g., CIDR [53]) or directly modeling the dropout phenomenon (e.g., ZIFA [54]). A different approach taken by other pipelines is to differentiate between true zeroes (gene not expressed) and zeroes arising from dropdown and dropout.

Many statistical algorithms developed for imputation recently show effectiveness in recovering dropouts. MAGIC [55], SAVER [56], and scImpute [57] are three leading methods in the field. MAGIC builds a Markov transition matrix constructed from a cell-to-cell similarity matrix. SAVER builds a Bayesian model to predict the dropouts based on the information of the observed genes. scImpute builds a regression model for each cell and imputes only genes with high dropout rates within a cell type by borrowing information from observed genes in that cell type. scImpute

requires running cell clustering first to detect cell types before building its imputation model. Most of these algorithms often demand expensive time and memory resources limiting their application on very large single cell datasets.

Another category of single cell imputation models are based on deep neural networks. DeepImpute [58] belongs to the category of predictive methods that uses a feed forward neural network to predict a missing gene expression in a cell given the observed values of genes that are highly correlated with that target gene. Recently, a group of methods were proposed to target that problem inspired from the emerging success of generative models. Among these methods are AutoImpute [59], scVI [60] and DCA [61]. scVI uses stochastic optimization and learns cell-specific embeddings using deep neural networks that best explain the observed data, whereas both AutoImpute and DCA uses autoencoders as their interior models. DCA encapsulates data denoising by learning the parameters of a zero-inflated negative binomial distribution using an autoencoder network. AutoImpute on the other hand trains an autoencoder to learn a latent representation of the data in lower dimension in hope that this representation will focus only on learning the important aspects of the dataset and hence reconstructing a denoised matrix correcting zeros if appropriate. However, both approaches have some limitations. DCA assumes that the dropout distribution follows a predetermined noise distribution which works well for simulated data, it is not clear if these assumptions work well in real data [62]. As for AutoImpute, it works on imputing only the top 1000 differential genes in the dataset, leaving the majority of the genes without imputation.

Here we present a novel approach, scGAIN (Figure 7.2), that tackles the single

cell imputation problem using Generative Adversarial Networks (GANs). GANs use adversarial learning to build a generative model of the data distribution [63–65]. Generative models are powerful tools that not only learn the distribution of data, but also generate synthesized data points that follow similar characteristics of the input data. Despite some recent efforts on utilizing the powerful capabilities of GANs into modeling scRNA-seq data [66, 67], they do not explore the usage of GANs to impute zero values in single cell data. As a result, the data distribution learned by these GANs, and thus data generated by them, will include the same dropout phenomenon that affects the input data and suffer from the same sparsity problem. In contrast, our proposed model modifies the standard GAN network architecture to accurately and efficiently impute missing gene expressions due to technical dropdown and dropouts in scRNA-seq data. Experimental results on simulated data show scGAIN can accurately impute zeroes and provide better estimates of true mean gene expression, and improves concordance between single cell and matched bulk datasets. scGAIN scales to large scRNA-seq datasets with thousands and millions of cells, which is infeasible for most existing statistical approaches. To our knowledge, this is the first work done on using GANs to impute scRNA-seq data.

7.3 Single Cell Generative Adversarial Imputation Nets (scGAIN)

7.3.1 Generative Adverserial Networks

A standard GAN [63] consists of two network components: Generator and Discriminator. The Generator network attempts to map samples from a low-dimensional

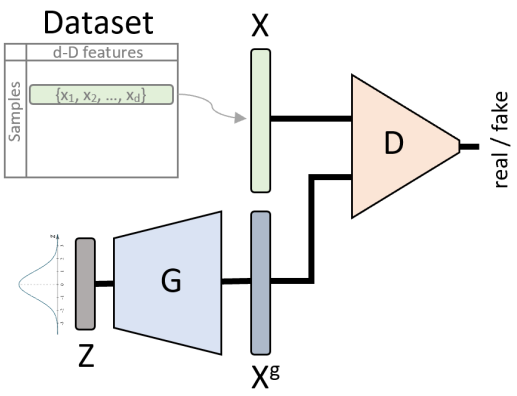

Figure 7.1: **Diagram of a standard GAN architecture.** Generator G takes low-dimensional random noise vector Z as input and generates a d-D sample X^g. Discriminator D takes either a real sample X or a fake sample X^g and outputs whether the input sample is real of fake.

high entropy distribution such as a Normal distribution, into a low-dimensional manifold within the high-dimensional space from which data is observed. The Discriminator network attempts to differentiate between samples from the real dataset and fake samples generated by the generator. Both components are trained jointly in an adversarial fashion where a min-max optimization problem forms a game between the generator and the discriminator. The generator's goal is to generate fake samples that are very close to the real samples such that the discriminator unable to distinguish between them, while the discriminator's goal is to learn the real data well that it would be able to accurately distinguish fake from real samples. When training converges, the generator would have reached a point that it can generate fake samples resembling the real distribution such that the well-trained discriminator is no longer able to distinguish between a sample present in the dataset and a sample generated by the generator.

Figure 7.1 shows a diagram of a standard GAN. Generator G takes low-

dimensional random noise vector $\mathbf{Z} = (z_1, z_2, ..., z_l); z_i \sim \mathcal{N}(0, 1)$ as input and generates a sample $\mathbf{X^g} = (x_1^g, x_2^g, ..., x_d^g)$. Discriminator D takes either a real sample $\mathbf{X}$ or a fake sample X^g and outputs whether the input sample is real of fake. A well-trained generator can generate random samples that follows the same distribution of $\mathbf{X}$ by drawing random values of $\mathbf{Z}$. The training of a GAN typically tries to solve the optimization problem

$$\min_G \max_D \mathbf{E}_X \log[D(X)] + \mathbf{E}_Z \log[1 - D(G(Z))] \tag{7.1}$$

7.3.2 GAIN Architecture and Model

In this section we briefly discuss the main changes that GAIN proposes. Figure 7.2 shows a diagram of the GAIN model we use for imputing dropouts in single cell datasets.

The Generator (G)

The main goal of the generative model in GAIN is to learn to infer the missing values in a sample $X = (x_1, x_2, .., x_k)$ for k features (genes) rather than focusing on generating the entire feature vector. The generator takes as part of the input a mask vector $M = (m_1, m_2, .., m_k); m_i \in \{0, 1\}$ dependent on X whereas $m_i = 0$ marks dropdown and dropout genes. The top part in figure 7.2 shows how the input to the generator is prepared. The vector $\overline{X}$ is prepared by filling the missing values in X with random noise values from noise vector Z. The generator takes a 2k-dimensional vector by concatenating $\overline{X}$ and M. The output layer in generator

network is a d-dimensional vector X^g where only the imputed values are of interest. By combining the imputed values from X^g with the rest of the observed values X we obtain $\tilde{X}$ which represent the generated sample from the generator and is fed to the discriminator as input. It should be noted that even though the values of X^g that doesn't correspond to dropout genes are replaced by the true observed values in X before given to the discriminator as input, the generator still gets penalized by the prediction error in those values as part of the generator's loss function optimized during training as a measure of how well the generator learned the observed part of the data.

The Discriminator (D)

Similar to the generator, the discriminator's model in GAIN is different from the standard GAN. The main difference is instead of having a single prediction of whether the entire input sample is real or fake, GAIN's discriminator gives separate prediction of whether each component $\tilde{x}_i \in \tilde{X}$ is real or fake given the entire sample. Since the discriminator outputs $\tilde{M}$ tends to have zeros for weakly imputed values and ones for true observed values, equivalently it tends to learn the mask vector M itself. Yoon et al. [68] showed theoretically in their paper that the discriminator needs as input what they called as a *Hint Mechanism*. A hint vector H must carry "enough" information about M, otherwise there would be multiple solutions of G that all can be optimal from the perspective of D. Whilst H should not be strictly equal to M otherwise the discriminator may trivially converge to H. As shown in figure 7.2 the hint component generates a small variation of M by setting a few randomly selected values in M to 0.5. Components of M with 0.5 values say nothing

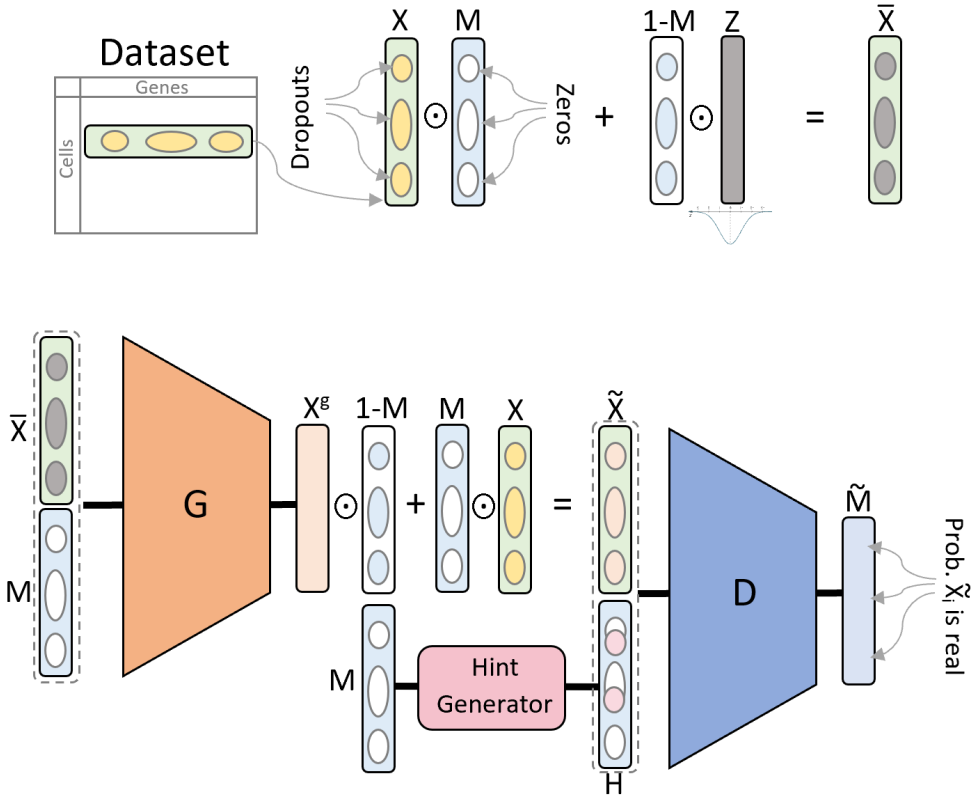

Figure 7.2: **An illustration of scGAIN's GAN architecture and input preparation for scRNA-seq imputation.** Top row describes how single cell dataset X is processed to produce input vector $\overline{X}$ to the scGAIN network. The binary mask vector M identifies which entries in the input matrix X are a target for imputation ($M_i = 0$ if $X_i = 0$). To create input vector $\overline{X}$, we fill imputation targets (where $M_i = 0$) with random noise Z_i, and fill the rest (where $M_i = 1$) with the original data X_i. The bottom row sketches the overall architecture of the scGAIN model. G is the GAN's generator: given input vector $\overline{X}$ and mask M it generates synthetic vector $\tilde{X}$ where imputation targets have been filled in. The hint generator produces hint vector H as a perturbed version mask M by setting small portions of values $M_i = 0$ to 0.5. D is the GAN's discriminator: it's task is to discriminate between the synthetic vector $\tilde{X}$ and hint H. Entries H are unknown to the discriminator D whether they were imputed or observed.

about their corresponding values in $\tilde{X}$ and its the discriminator job to classify.

Objective Function

Along with the changes in the generator and discriminator models from a standard GAN, the loss functions are also slightly different. First, the discriminator no longer takes two sets of samples (real or fake). But rather for a given (imputed) sample, the discriminator should distinguish between observed and imputed values of that sample. Consequently, the discriminator's objective function would be for a given G:

$$\begin{aligned} \max_{D} \mathbf{E}_{\tilde{X},M,H}\{M \odot log[D(\tilde{X},H)]+ \\ (1-M) \odot log[1-D(\tilde{X},H)]\} \end{aligned} \tag{7.2}$$

While the generator's objective function would be for a given D:

$$\begin{aligned} \min_{G} : \mathbf{E}_{X,Z,M,H}\{(1-M) \odot log[1-D(\tilde{X},H)]\}+ \\ \lambda \mathbf{E}_{X,Z,M}[M \odot X - M \odot G(\overline{X},M)]^2 \end{aligned} \tag{7.3}$$

The first term in the loss function represents the cross entropy of the imputed values of $\tilde{X}$. The second regularized term reflects a mean squared error (MSE) of the observed values of $\overline{X}$.

Imputation Mask Generation

During either training or imputation steps, a mask matrix is required to iden-

tify which entries in the input matrix are a target for imputation. The mask marks if an entry in the data matrix is missing, unless it belongs to a gene of zero or significantly low mean expression determined by a parameter threshold θ. Therefore, the mask value of gene i in cell j is determined as follows:

$$m_{ij} = \begin{cases} 0, & \text{if } x_{ij} = 0 \text{ and } \overline{x_i} > \theta. \\ 1, & \text{otherwise.} \end{cases} \tag{7.4}$$

where $\overline{x_i}$ is the mean expression of gene i calculated from non-zero values. Note that the threshold θ aims at keeping genes of true zeros out of the scope of imputation.

7.3.3 scGAIN Stratified Training

Throughout our experiments, we noticed that the original GAIN [68] training was not stable when directly applied to single cell data. In single cell, data is much sparser (samples can have up to 95% zeros) compared to the standard application of GAIN in image analysis (usually tested against up to 50% sparsity). In addition, in single cell data, as opposed to image data, sparsity depends on values of the observed data. Figure 7.3 shows the distribution of zeros in each gene with respect to its mean expression calculated over non-zero observations in the data. Genes with lower mean expressions are more prone to dropdowns.

During training, since most zeros in a sample belong to genes with low mean expression, dropouts in genes of high expression (which are rare events) were poorly

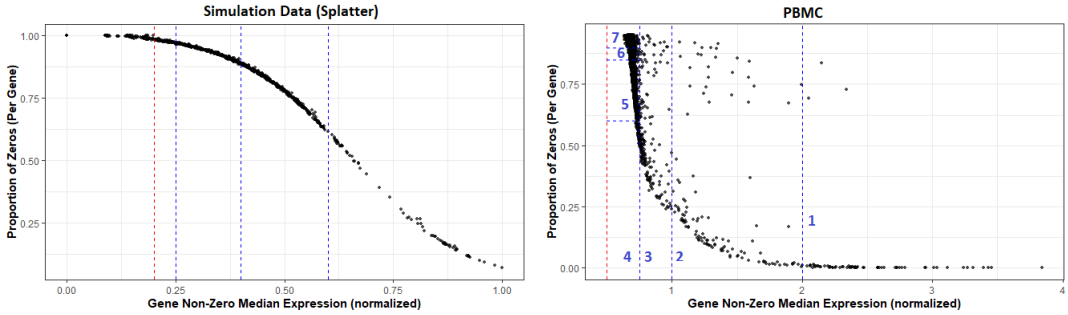

Figure 7.3: **Distribution of zeros in each gene with respect to its mean expression** calculated over non-zero observations in simulation data from figure 7.4 **(Left)** and PBMC **(Right)**. Vertical dotted lines represent the different stratification thresholds used during the training on each dataset. Red dotted lines represent the zero cutoff, genes beyond that point were not attempted for imputation. In PBMC, both horizontal and vertical cutoffs are used to stratify the really dense area with low mean expressions, resulting in 7 stratification rounds.

trained. These genes are rarely selected by the hint mechanism to contribute to the discriminator's loss function, resulting in a generator with poor accuracy to impute genes with rare dropout events. To remedy that issue, we stratify the training process by applying an expression threshold to the genes that may be to be masked for imputation. For example, during the first few epochs, the model trains to impute only genes of high mean expression (i.e., dropouts), that controls the number of zeros in the training mask and gives genes with rare sparsity to be learned first (section imputation mask generation). Then gradually the threshold is reduced to include more and more genes with higher sparsity rates (i.e., dropdowns). Figure 7.3 shows vertical lines with different stratification thresholds used during training on simulated dataset from figure 7.2.

7.3.4 scGAIN Architecture and Parameter Configuration

Both the generator and the discriminator networks have similar network structure. Both consist of an input layer of size $2k$ (k is the number of genes), then two hidden fully connected layers of 256 and 128 nodes each. Then an output layer of size k with a sigmoid activation function to ensure that the output values are in the range $[0, 1]$. Training is done alternatively between the generator and the discriminator in a fusion of mini-batches of 128 cells at a time. Unless mentioned otherwise, the hint generator would randomly select 10% of the mask vector H to mask its values to 0.5. We chose $\lambda = 10$ as the regularization parameter of MSE term in the loss function of the generator.

7.3.5 Data Normalization

Before data is input to our model, the raw count matrix undergoes several steps of pre-processing. First, the counts are normalized by the total number of UMIs to account for different sequencing depth between cells. Then the normalized data is transformed into log scale similar to the normalization procedure described by Zhang et al. [69]. Finally, each cell expression vector is divided by the maximum gene expression present in that cell to ensure that the network takes input values in the range $[0, 1]$.

7.4 Datasets

7.4.1 Simulated Datasets using Splatter

We used Splatter's R/Bioconductor package [62] to generate two simulated datasets containing three and six cell type groups. In both, we simulated 1000 genes for 20,000 cells. We generated cells such that the proportions the group are 40%, 30% and 30% in the three groups case, and 30%, 20%, 10%, 20%, 10% and 10% in the six groups case. We used parameters $dropout.shape = -1$, $dropout.mid = 3.2$ and 5 in R function *splatSimulate()* to add 65% and 85% dropouts to the count matrix, respectively.

7.4.2 SimData60k

In this dataset, we simulated the gene expression matrix of 2000 genes for three cell types. We recreated the simulation design and used the same parameters suggested by [57]. log10-transformed read counts were directly generated in the following configuration. For each gene, its mean expression is drawn from a Normal distribution $\mathcal{N}(1.8, 0.5)$ and its standard deviation is drawn from a Normal distribution $\mathcal{N}(0.6, 0.1)$. Then 900 genes were randomly selected to be differential across cell types; each cell type has different 300 genes showing higher expression levels in that cell types than the other two types. We then generate 20,000 cells per cell type using gene means and standard deviations as calculated earlier for each type. Then dropout genes are introduced into both datasets. Dropout rates are calculated

using a double exponential function of each gene's mean expression. Finally, zeros are placed into the expression matrix using a Bernoulli distribution based on the calculated dropout rates.

7.4.3 PBMC Dataset

This is a dataset of approximately 68,000 cells of peripheral blood mononuclear cells collected from a healthy donor [69]. This dataset is provided publicly on 10x Genomics website.

The data is pre-processed to detect dropouts from genes that consistently show zero expression. This is done by filtering out genes that are zero in 99% of the cells. Then a subset of cells was used as training data in our scGAIN model. To guarantee that the network is trained on samples with enough data rather than outliers, significantly sparse cells that have less than 2% non-zero genes were excluded from the training data. That leaves a subset of approximately 18,000 cells used as training data to impute about 3000 genes. It should be noted that this filtering criteria is not mandatory. scGAIN can still accept the full normalized and log-transformed matrix as its input.

7.5 scGAIN imputes zeros without adding biases in simulated data

To study the effectiveness of imputation using scGAIN in a controlled setting, we applied scGAIN to simulated scRNA-seq data generated using Splatter ([62]) with sparsity levels of 85% (see Section Simulated Dataset using Splatter for data

details) and compared it's performance with representative state-of-the-art imputation methods.

Figure 7.4 summarizes results of this comparison. We compare performance in four different aspects. First, imputation accuracy: we plot all missing values after imputation versus the true values (gene expression without dropouts). We summarize the plot by calculating two metrics: pearson correlation and mean square error (MSE). scGAIN shows best performance in both metrics: 0.94 correlation vs. DrImpute's 0.88 which was second highest, and 0.005 MSE vs. DrImpute's 0.035 which was second lowest. Second, clustering cell types: we show the effect of imputation in recovering underlying structure of the different cell types, through calculating t-distributed stochastic neighbor embedding (t-SNE) of the true data, raw data without any imputation, and data after imputation by each method. scGAIN and DrImpute successfully recovered the clustering of cells which was relatively distorted in the raw data due to dropouts. Third, t-SNE plots of the same cells using 100 genes that are not differentially expressed in the simulated (no dropout) data. These group of genes remain not differentially expressed in scGAIN, whereas methods like DrImpute and Magic showed undesired clustering introducing false signals into the data. In summary, even though the overall clustering of a method like DrImpute may look slightly better, the formation of new clusters in truly non-differential genes suggest against its use for real data. Lastly, heatmap of expression for the top 50 variable genes in the true data. With dropouts, the heatmap is distorted with an excessive number of low-to-zero values across the entire heatmap, whereas scGAIN and DrImpute produces the closest heatmap to the truth with most dropouts recovered.

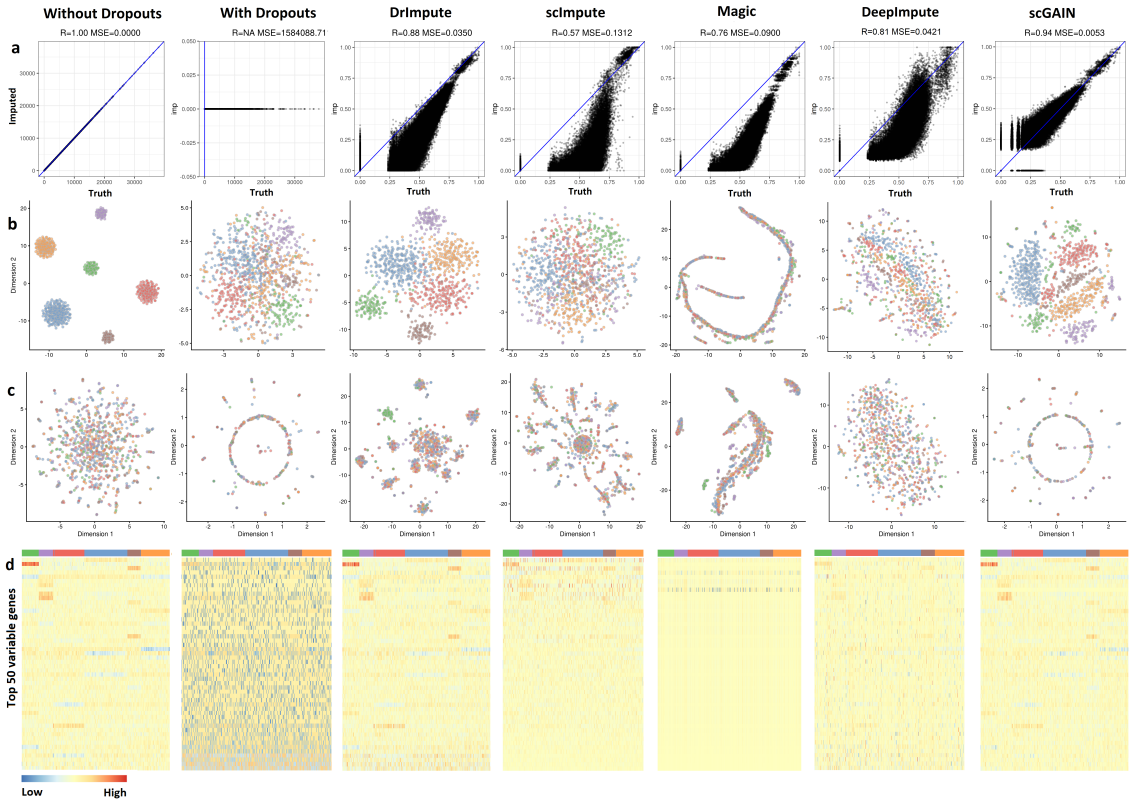

Figure 7.4: **Overall performance comparison on simulated data.** Performance of five imputation methods (DrImpute, scImpute, Magic, Viper, scGAIN) on simulated dataset (ncells = 20,000, ngenes = 1000) containing six cell types with sparsity 85%. **a)** Scatter plots of imputed vs. (non-dropout) simulated data showing the correlation and MSE measures between the truth and imputed values. scGAIN shows best performance in both metrics **b)** t-SNE visualization of random 1000 cells using the 100 most variable genes from each method. scGAIN and DrImpute successfully recovered the clustering of cells which was distorted in the simulated data with the dropouts. **c)** t-SNE visualization of 100 non-differentially expressed genes. These group of genes remain non-differentially expressed after imputation with scGAIN, whereas methods like DrImpute and Magic showed undesired clustering introducing false signals into the data. **d)** Heatmap of the top 50 variable genes between cell types in the full data. scGAIN and DrImpute generate the closest heatmaps to the simulated data.

7.6 scGAIN produces stable imputations around the true mean expression with lower variance than other methods

We compare three imputation tools, each representing a category of imputation approaches, on a simulated data (see Section Simulated Data). We chose scImpute as a statistical decomposition method, DeepImpute as a prediction method that uses feed-forward neural, and our approach (scGAIN) as a generative method. DeepImpute works as a point estimator of each imputed value. scGAIN on the other hand as a generative model it can give multiple draws of possible imputed values of the missing values based on the learnt distribution observed in the population. We did not include DrImpute in this comparison as it was not able to scale to data of this size. One way to illustrate this is to observe how the individual imputed values are distributed around the true mean expression of the gene in a certain cell type. Figure 7.5 shows how the individual gene expressions in the imputed matrix (Y-axis) are compared to the corresponding true mean expression (X-axis) used in the simulation. scGAIN shows more stable point estimates around the simulation means compared to the other two methods. Decreased variablity (around precise imputated values) is desirable in real experiments since it may increase power of downstream statisitcal analyses. scGAIN consistently gives accurate estimates corresponding to true means even at low expression levels whereas DeepImpute tends to overestimate low and moderate-level expressions.

Figure 7.5: **Bias and variance comparison of imputed data.** For three imputation methods: scGAIN, scImpute, DeepImpute on simulated data, we plot the individual gene expressions in the imputed matrix obtained from each method are compared to the corresponding true mean expression used in the simulation. **(Left)** Box plot of absolute imputation error for each mean expression bin. True mean expressions are divided into bins with equal number of genes. **(Right)** Scatter plot of imputed gene expressions in different cells of one type versus the true mean expression of that gene. scGAIN shows accurate estimates around the true mean with smaller variance compared to the other two methods. DeepImpute tends to overestimate expression in genes with low and moderate-level expression.

7.7 scGAIN increases correspondence with matched bulk expression of marker genes in PBMC dataset

To evaluate scGAIN on real data and compare its imputation with other methods, we used the PBMC [69] dataset to study the effect of imputation on a set of marker genes for CD4+ vs CD8+ T-cells. First, we determined genes that are differentially expressed between the two cell types from a bulk RNA-seq cell-sorted dataset [70] as the gold standard (by running DESeq [13] and selecting genes with p-value < 0.05). Since not all imputation methods are scalable to impute PBMC in reasonable time, we only include DCA, DeepImpute and MAGIC as a subset for comparison in this analysis. Figure 7.6a) shows expression of seven marker genes

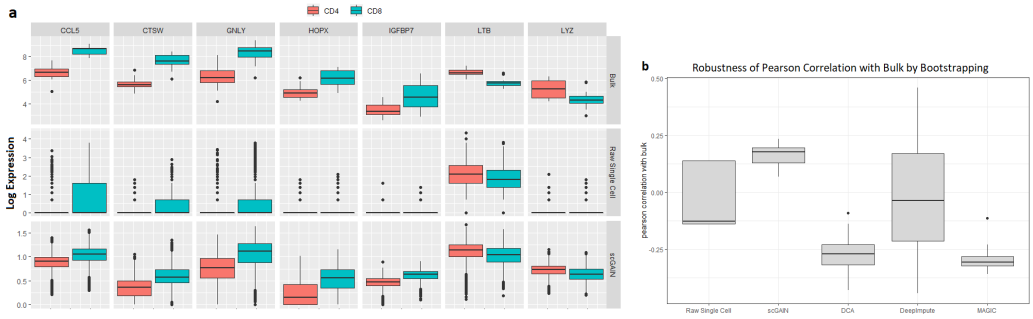

Figure 7.6: **scGAIN increases correspondence with matched bulk expressions of marker genes in PBMC dataset. a)** shows expression of seven marker genes that distinguish between CD4+ and CD8+ cells in bulk, and how their expression levels look like in raw single cell data and data imputed by scGAIN. **b)** shows that scGAIN produces the most robust imputation and the highest correlation among the methods.

that distinguish between CD4+ and CD8+ cells in bulk, and how their expression levels look like in raw single cell data and data imputed by scGAIN. ScGAIN recovers some of the differential expression in these genes which is lost when computing differential expression on the non-imputed data.

Next we wanted to measure the robustness of that correspondence for the top 50 differentially expressed genes based on the bulk data with the imputed single-cell data from each method. Robustness is measured by generating 100 bootstrap single-cell samples for 100 random cells, and then computing the Pearson correlation of the difference in log median expression for the two celltypes between the imputed single-cell data and the bulk data for each bootstrap sample. Figure 7.6b) shows that scGAIN produces the most robustness and the highest correlation among the methods.

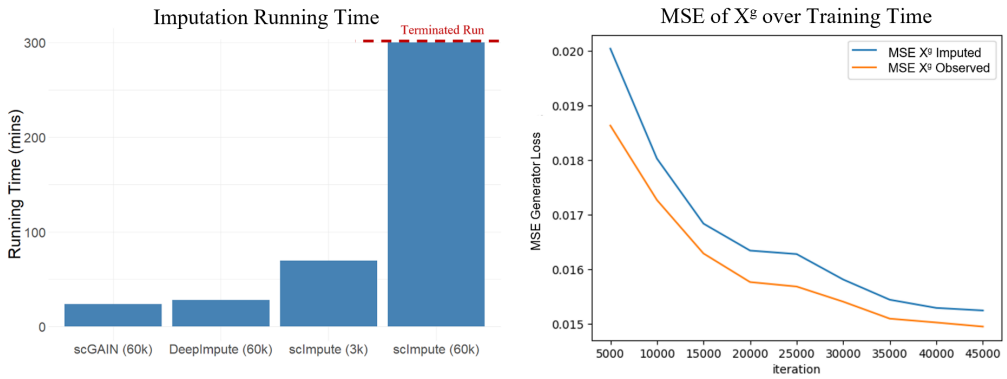

Figure 7.7: **scGAIN scales to large datasets.** Imputation Running Time for 60,000 cells of Simulated Data (simData60K) used by the three methods. scImpute has two columns: one for imputing a small subset of the dataset of only 3,000 cells, and another for the full dataset. scImpute's run over the entire 60,000 cells was terminated after running for a full day without finishing.

7.8 scGAIN is efficient and scalable to large scRNA-seq experiments

Since scGAIN uses neural network models, it scales well to efficiently deal with large datasets which provide enough samples for the network parameters to converge. SimData60k is a simulated dataset with 60,000 cells (see Simulated Data). Figure 7.7 summarizes the running time used by each of scGAIN, DeepImpute and scImpute. scGAIN and DeepImpute both have a significant advantage over statistical approaches like scImpute which needs to process the full dataset at once. For instance, scImpute requires calculation of pairwise cell distances to perform clustering as an early step of its pipeline which can be an intense time and memory consuming process.

Chapter 8: Discussion and Conclusion

The work discussed in this thesis presents a line of research that addresses the use of graph representation and algorithms in a variety of genomic analyses. Specifically, we introduced the concept of transcriptome segmentation and formulating the definitions of segments and segment counts as summarized statistics of RNA-seq experiments that preserve a localized measure across the genome. Then we illustrated how our framework can be used to run analyses on the three resolutions of RNA-seq: i.e. Gene-level, Transcript-level and Alternative Splicing-level analyses. We showed how our approach can empower lightweight, ultra-fast pseudo-alignment tools like Kallisto or Salmon with capabilities to provide alternative splicing measures that achieves comparable accuracy to count-based approaches while maintaining the speed and efficiency of such tools. Then we tackled the problem of transcript quantification under incomplete annotation and propose a segment-based correction procedure to reduce the bias in the estimated abundances present in such scenarios, without the need to assemble the missing transcripts first. After that, we explored the possibility to Bridge the gap between linear and graph alignment over whole genome population references and evaluating its benefits on handling allelic variations of highly polymorphic genes like HLA genes. Lastly, we proposed a

generative model approach to impute single cell RNA-seq data to correctly recover the missing values of dropout genes by adopting deep GANs for that purpose.

The benefits of our work vary from providing faster and more efficient pipelines, empowering available approaches with more capabilities or introducing new perspectives into existing challenges. In this Chapter we discuss the implications of the work presented in this dissertation, place it in context of existing work, and give pointers to future work.

8.1 Yanagi is a fast and interpretable segment-based approach for gene, transcript and alternative splicing level analysis:

In this work, we have formalized the concept of transcriptome segmentation and proposed an efficient algorithm for generating segment libraries from transcript libraries based on a length parameter L (typically chosen dependent on an experiment-specific RNA-seq library construction). The resulting segment sequences are used with pseudo-alignment tools to quantify expression at the segment level, providing enough information for a variety of expression analyses. We have characterized segment libraries for the reference transcriptomes of *Drosophila melanogaster* and *Homo sapiens* for various read-length RNA-seq experimental designs. We also provide a novel gene-level visualization of transcriptome segments and transcript structure for ease of interpretation. Finally, we have demonstrated the use of segment-level quantification in differential gene expression and alternative splicing analysis.

Using a segment library rather than the standard transcriptome succeeds in significantly reducing ambiguous alignments where reads are multi-mapped to several sequences in the reference, thereby decoupling the pseudo-alignment and quantification steps used in current k-mer based pipelines for gene expression analysis. Moreover, using segment counts as statistics for gene-level differential expression, transcript quantification and alternative splicing analyses achieves performance comparable to counting-based approaches (e.g. rMATS for splicing analysis) while using fast and lightweight pseudo-alignment. The notion of transcript segmentation as introduced here and implemented in Yanagi has the potential to extend the application of lightweight, ultra-fast, pseudo-alignment algorithms to a wider variety of RNA-seq analyses.

We discussed at the end of each chapter how our completed work can be further extended in the future. One extension is to adjust our models to handle the different sources of biases seen in RNA-seq data. We focused on one source that is usually being ignored by most present tools; the bias due to incomplete annotation. The preliminary results shown here provides a positive proof of concept. However, more elaborate experiments are essential to have a better understanding of the various aspects of the problem and how our proposed work handle them.

8.2 Linearizing whole genome population reference graph for certain genes combines the advantages of both linear and graph-based alignment:

We proposed an approach that takes advantage of representing population haplotypes as a graph, and efficiently linearizes the graph through segmentation model in Yanagi. Our method generates a set of maximal L-disjoint segments representing the linearized population graph into a reference sequence library that can be used by any alt-aware linear aligner. Using segments empowers any linear aligner with the efficient graph representation of population variations, while avoiding the expensive computational overhead of aligning over graphs. We tested our approach on the highly polymorphic HLA genes which have significant medical importance.

Preliminary results showed that we can achieve comparable performance to graph aligners using linear aligners assisted with population segments without compromising their space and computational requirements. However, experiments were only evaluated on the read extract of HLA reads. A future extension to that approach would be to perform HLA typing and comparing its results. Another extension would be to examine developing a similar approach based only on the list of variations, e.g. provided by the *1000 Genomes* project. Without the multi-sequence alignment being provided as input, it would be interesting challenge to see how practical our approach would be.

8.3 GAN-based approaches can be efficient and successful in imputing single cell RNA-seq data:

Successful use of imputation in a single-cell expression analysis requires methods that can perform stable and accurate imputation, that recapitulates underlying features of the cell populations that would be observed with higher throughput (e.g., from cell-sorted bulk RNA-seq), without introducing additional bias or structure in the data. Furthermore, these methods need to scale to large datasets. We have shown that while architectures like DrImpute and DeepImpute are scalable and can perform accurate imputation, their stability can suffer, and in the case of DrImpute introduces artificial structure in the data. scGAIN showed similar scalability while improving the stability of these methods and without introducing artificial structure in the data.

scGAIN uses GANs as the core method to learn a data distribution from which imputation can be performed. In their standard formulation, GANs learn the distribution of input data using a deep network capable of generating samples that resemble the input data. Implicitly, the generated samples have the same characteristics of the learned data i.e. if the input data is sparse, the generator will also generate sparse samples. So, implementing a standard GAN model will not solve the imputation problem. Recent efforts have established the use of GANs to impute missing data in vision and image processing [68, 71]. The key idea in these methods is to use GANs to learn the structure of missing data rather than

the structure of the entire data space. This makes the GAN more focused, effective and speeds up model convergence. We adopt the GAIN model discussed in [68], introduced to impute missing pixels of images.

Imputing dropdowns and dropouts in scRNA-seq data can be more challenging than the vision case. The amount of missing data is significantly large in single cell RNA-seq data compared to the training data used in vision. In addition, missing data in the single-cell case occurs as a function of the missing values (low expression) whereas this is not necessarily the case in vision problems. Moreover, the evaluation metric used in vision is quite different. When imputing portions of an image, the accuracy of prediction of individual pixels is not that important as long as the final imputed image looks coherent and realistic. While in scRNA-seq single value predictions of individual genes are essential. This increases the complexity of the model when adopting imputation algorithms from vision to our domain of interest. We overcame some of these issues by developing a novel method to create an imputation mask based on the values of the non-missing data, i.e., the observed expression of genes across cells. Also, we used a stratified training strategy for scGAIN that addresses these issues as well.

While scGAIN has shown accuracy, stability and scalability without introducing artificial structure to imputed data, there are several ways in which this approach can be improved. First, a limitation of GANs in general, since they are non-parametric methods, we are unable to provide a statement of uncertainty around imputed values that could be used in downstream analyses. Recent work showing the connection between GANs a probabilistic structure can be used to alleviate this

problem. Second, while the method used here to produce a mask vector produced satisfactory results, ideas from parametric models of dropouts may be useful to generate a more accurate mask. Finally, the GAIN architecture requires the generation of a hint vector as a reference from which to evaluate the GAN's discriminator's performance. A revised architecture that removes this dependence on a hint generator would be preferable.

Appendix A: Transcriptome Segmentation Algorithm

Algorithm 1 Yanagi's Segments Library Generation

Require: Transcriptome Annotation (GTF File), Transcripts Sequences (FASTA Files)

$TxDB \leftarrow makeTxDBFromGTFFile$ ▷ Preprocessing Step
$DExs \leftarrow disjoinExons(TxDB)$
$TxDB \leftarrow adjustTxDB(TxDB, DExs)$

Step 2 - Segment Graph Construction

procedure CONSTRUCT_SEG_GRAPH($TxDB, g, L$) ▷ SgG of gene g
    $G \leftarrow emptygraph$
    $St \leftarrow \phi$
    $prev \leftarrow$ DUMMY_NODE
    for each $Tx \in \mathcal{T}xDB(g)$ **do** ▷ For each transcript in gene g
        $loc \leftarrow start(Tx)$
        while $loc < end(Tx)$ **do** ▷ Iterate till the end of transcript
            $gr \leftarrow GenomicRange(Tx, loc, loc + L)$
            $Exs \leftarrow exons[TxDB(gr)]$
            $w, loc_{new} \leftarrow$ REFINE_NODE(Exs, loc, L) ▷ Node refinement step
            $node \leftarrow getOrCreateNode(G, < Exs, loc, w >)$
            $Txs(node) \leftarrow Txs(node) + Tx$
            $Next(prev) \leftarrow Next(prev) + node$ ▷ Make an edge
            if $Txs(prev) \neq Txs(node)$ **then** ▷ Mark branches in graph
                for each $n \in Next(prev)$ **do**
                    $St \leftarrow St + n$
            $prev \leftarrow node$ ▷ Advance loop
            $loc \leftarrow loc_{new}$
    return G, St ▷ The SgG of gene g

Step 3 - Segments Library Generation

procedure GENERATE_SEGMENTS(G, St)
    for each $node \in St$ **do** ▷ Iterate over branching points in G
        $seg \leftarrow newsegment$ ▷ Initialize a new segment
        $seg.appendNode(node)$
        while $|Next(node)| = 1$ **do** ▷ Aggregating chain of nodes into a segment
            $node \leftarrow Next(node)$
            $seg.appendNode(node)$
        $outputSegment(seg)$

Appendix B: Homogenous Cell Population from PBMC are Zero-Inflated

Recent work argues that droplet scRNA-seq data, for example the PBMC data, is not zero-inflated when considering technical replicates [51]. This would imply observed zeros are the result of dropdown and cell-type heterogeneity. In order to confirm that the observed zero-inflation in single cell raw data is not strictly the result of having heterogenous population of cells, we subset one subpopulation (CD19+ B) from PBMC. Figure B.1 shows that zero-inflation fits data better even when looking at a homogenous cell population of CD19+ B cells.

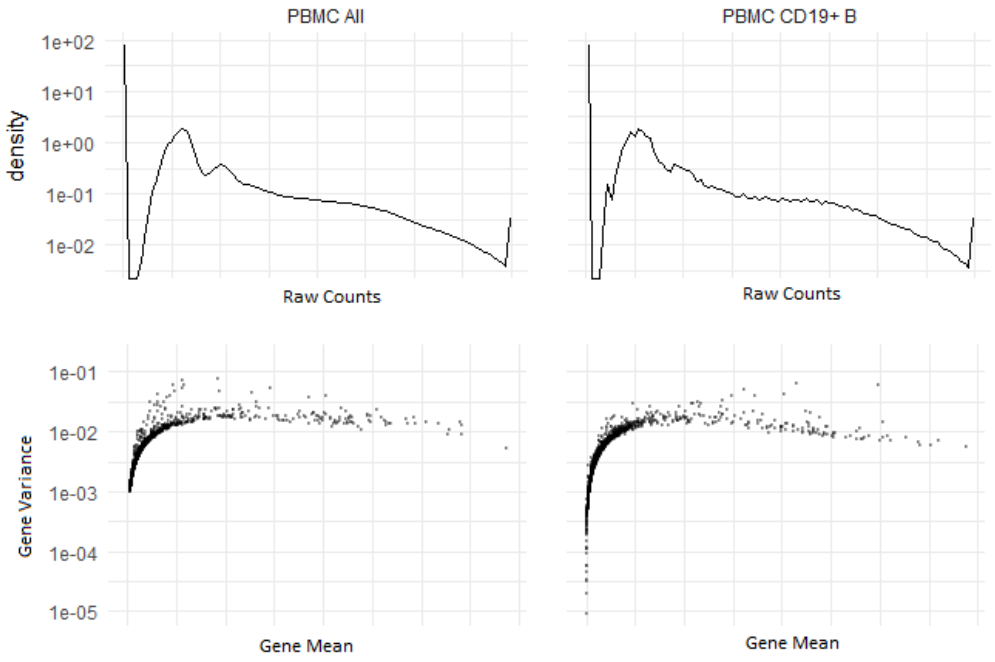

Figure B.1: Plot showing the distribution of raw counts and mean-variance expressions of genes from PBMC dataset. It shows that the data is zero-inflated with heterogenous cell population (PBMC All) as well as homogenous cell population (PBMC CD19+ B).

Bibliography

[1] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature Methods*, 9(4):357–359, 2012.

[2] Cole Trapnell, Lior Pachter, and Steven L Salzberg. Tophat: discovering splice junctions with rna-seq. *Bioinformatics*, 25(9):1105–1111, 2009.

[3] Daehwan Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley, and Steven L Salzberg. Tophat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biology*, 14(4):R36, 2013.

[4] Cole Trapnell, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J Van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript assembly and quantification by rna-seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, 2010.

[5] Rob Patro, Stephen M. Mount, and Carl Kingsford. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nature Biotechnology*, 32(5):462–464, May 2014.

[6] Avi Srivastava, Hirak Sarkar, Nitish Gupta, and Rob Patro. Rapmap: a rapid, sensitive and accurate tool for mapping rna-seq reads to transcriptomes. *Bioinformatics*, 32(12):i192, 2016.

[7] Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic rna-seq quantification. *Nature Biotechnology*, 34(5):525–527, 2016.

[8] Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*, 2017.

[9] Bo Li and Colin N Dewey. Rsem: accurate transcript quantification from rna-seq data with or without a reference genome. *BMC Bioinformatics*, 12(1):323, 2011.

[10] S Lawrence Zipursky, Woj M Wojtowicz, and Daisuke Hattori. Got diversity? wiring the fly brain with dscam. *Trends in Biochemical Sciences*, 31(10):581–588, 2006.

[11] Hagen Tilgner, Fereshteh Jahanbani, Tim Blauwkamp, Ali Moshrefi, Erich Jaeger, Feng Chen, Itamar Harel, Carlos D Bustamante, Morten Rasmussen, and Michael P Snyder. Comprehensive transcriptome analysis using synthetic long-read sequencing reveals molecular co-association of distant splicing events. *Nature Biotechnology*, 33(7):736–742, 2015.

[12] Brian J Haas, Arthur L Delcher, Stephen M Mount, Jennifer R Wortman, Roger K Smith Jr, Linda I Hannick, Rama Maiti, Catherine M Ronning, Douglas B Rusch, Christopher D Town, et al. Improving the arabidopsis genome annotation using maximal transcript alignment assemblies. *Nucleic Acids Research*, 31(19):5654–5666, 2003.

[13] Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome Biology*, 15(12):550, 2014.

[14] Davis J. McCarthy, Yunshun Chen, and Gordon K. Smyth. Differential expression analysis of multifactor rna-seq experiments with respect to biological variation. *Nucleic Acids Research*, 40(10):4288–4297, 2012.

[15] Cole Trapnell, David G Hendrickson, Martin Sauvageau, Loyal Goff, John L Rinn, and Lior Pachter. Differential analysis of gene regulation at transcript resolution with rna-seq. *Nature Biotechnology*, 31(1):46, 2013.

[16] Kimon Froussios, Kira Mourão, Nicholas J Schurch, and Geoffrey J Barton. Identifying differential isoform abundance with rats: a universal tool and a warning. *bioRxiv*, 2017.

[17] Michael I Love, John B Hogenesch, and Rafael A Irizarry. Modeling of rna-seq fragment sequence bias reduces systematic errors in transcript abundance estimation. *Nature Biotechnology*, 34(12):1287, 2016.

[18] Sebastian Schafer, Kui Miao, Craig C Benson, Matthias Heinig, Stuart A Cook, and Norbert Hubner. Alternative splicing signatures in rna-seq data: Percent spliced in (psi). *Current Protocols in Human Genetics*, pages 11–16, 2015.

[19] Ruolin Liu, Ann E Loraine, and Julie A Dickerson. Comparisons of computational methods for differential alternative splicing detection using rna-seq in plant systems. *BMC Bioinformatics*, 15(1):364, 2014.

[20] Simon Anders, Alejandro Reyes, and Wolfgang Huber. Detecting differential usage of exons from rna-seq data. *Genome Research*, 22(10):2008–2017, 2012.

[21] Shihao Shen, Juw Won Park, Zhi-xiang Lu, Lan Lin, Michael D Henry, Ying Nian Wu, Qing Zhou, and Yi Xing. rmats: robust and flexible detection of differential alternative splicing from replicate rna-seq data. *Proceedings of the National Academy of Sciences*, 111(51):E5593–E5601, 2014.

[22] Jorge Vaquero-Garcia, Alejandro Barrera, Matthew R. Gazzara, Juan Gonzalez-Vallinas, Nicholas F. Lahens, John B. Hogenesch, Kristen W. Lynch, Yoseph Barash, and Juan Valcárcel. A new view of transcriptome complexity and regulation through the lens of local splicing variations. *eLife*, 5:e11752+, February 2016.

[23] Yin Hu, Yan Huang, Ying Du, Christian F Orellana, Darshan Singh, Amy R Johnson, Anaïs Monroy, Pei-Fen Kuan, Scott M Hammond, Liza Makowski, et al. Diffsplice: the genome-wide detection of differential splicing events with rna-seq. *Nucleic Acids Research*, 41(2):e39–e39, 2012.

[24] Gael P Alamancos, Amadís Pagès, Juan L Trincado, Nicolás Bellora, and Eduardo Eyras. Leveraging transcript quantification for fast computation of alternative splicing profiles. *Rna*, 21(9):1521–1531, 2015.

[25] Juan L Trincado, Juan C Entizne, Gerald Hysenaj, Babita Singh, Miha Skalic, David J Elliott, and Eduardo Eyras. Suppa2: fast, accurate, and uncertainty-aware differential splicing analysis across multiple conditions. *Genome Biology*, 19(1):40, 2018.

[26] Nicholas F Lahens, Ibrahim Halil Kavakli, Ray Zhang, Katharina Hayer, Michael B Black, Hannah Dueck, Angel Pizarro, Junhyong Kim, Rafael Irizarry, Russell S Thomas, et al. Ivt-seq reveals extreme bias in rna sequencing. *Genome Biology*, 15(6):R86, 2014.

[27] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.

[28] 1000 Genomes Project Consortium et al. A global reference for human genetic variation. *Nature*, 526(7571):68, 2015.

[29] James Robinson, Jason A Halliwell, James D Hayhurst, Paul Flicek, Peter Parham, and Steven GE Marsh. The ipd and imgt/hla database: allele variant databases. *Nucleic Acids Research*, 43(D1):D423–D431, 2014.

[30] Deanna M Church, Valerie A Schneider, Tina Graves, Katherine Auger, Fiona Cunningham, Nathan Bouk, Hsiu-Chuan Chen, Richa Agarwala, William M McLaren, Graham RS Ritchie, et al. Modernizing reference genome assemblies. *PLoS Biology*, 9(7):e1001091, 2011.

[31] 1000 Genomes Project Consortium et al. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061, 2010.

[32] Daehwan Kim, Joseph M Paggi, and Steven Salzberg. Hisat-genotype: Next generation genomic analysis platform on a personal computer. *bioRxiv*, page 266197, 2018.

[33] Hannes P Eggertsson, Hakon Jonsson, Snaedis Kristmundsdottir, Eirikur Hjartarson, Birte Kehr, Gisli Masson, Florian Zink, Kristjan E Hjorleifsson, Aslaug Jonasdottir, Adalbjorg Jonasdottir, et al. Graphtyper enables population-scale genotyping using pangenome graphs. *Nature Genetics*, 49(11):1654, 2017.

[34] Alexander T Dilthey, Pierre-Antoine Gourraud, Alexander J Mentzer, Nezih Cereb, Zamin Iqbal, and Gil McVean. High-accuracy hla type inference from whole-genome sequencing data using population reference graphs. *PLoS Computational Biology*, 12(10):e1005151, 2016.

[35] Benedict Paten, Adam M Novak, Jordan M Eizenga, and Erik Garrison. Genome graphs and the evolution of genome inference. *Genome Research*, 27(5):665–676, 2017.

[36] Mohamed K. Gunady, Steffen Cornwell, Stephen M. Mount, and Héctor Corrada Bravo. Yanagi: Transcript Segment Library Construction for RNA-Seq Quantification. In Russell Schwartz and Knut Reinert, editors, *17th International Workshop on Algorithms in Bioinformatics (WABI 2017)*, volume 88 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[37] Mohamed K Gunady, Stephen M Mount, and Hector Corrada Bravo. Yanagi: Fast and interpretable segment-based alternative splicing and gene expression analysis. *BMC Bioinformatics*, 20(1):1–19, 2019.

[38] Charlotte Soneson, Katarina L Matthes, Malgorzata Nowicka, Charity W Law, and Mark D Robinson. Isoform prefiltering improves performance of count-based methods for analysis of differential transcript usage. *Genome Biology*, 17(1):12, 2016.

[39] Alyssa C Frazee, Andrew E Jaffe, Ben Langmead, and Jeffrey T Leek. Polyester: simulating rna-seq datasets with differential transcript expression. *Bioinformatics*, 31(17):2778–2784, 2015.

[40] Lynn Yi, Harold Pimentel, Nicolas L Bray, and Lior Pachter. Gene-level differential analysis at transcript-level resolution. *Genome Biology*, 19(1):53, 2018.

[41] Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. Star: ultrafast universal rna-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.

[42] Charlotte Soneson, Michael I Love, and Mark D Robinson. Differential analyses for rna-seq: transcript-level estimates improve gene-level inferences. *F1000Research*, 4, 2015.

[43] Gordon K Smyth et al. Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Stat Appl Genet Mol Biol*, 3(1):3, 2004.

[44] Charity W Law, Yunshun Chen, Wei Shi, and Gordon K Smyth. Voom: precision weights unlock linear model analysis tools for rna-seq read counts. *Genome Biology*, 15(2):R29, 2014.

[45] Timothy Sterne-Weiler, Robert J. Weatheritt, Andrew J. Best, Kevin C.H. Ha, and Benjamin J. Blencowe. Efficient and accurate quantitative profiling of alternative splicing patterns of any complexity on a laptop. *Molecular Cell*, 72(1):187 – 200.e6, 2018.

[46] Luca Denti, Raffaella Rizzi, Stefano Beretta, Gianluca Della Vedova, Marco Previtali, and Paola Bonizzoni. Asgal: aligning rna-seq data to a splicing graph to detect novel alternative splicing events. *BMC Bioinformatics*, 19(1):444, Nov 2018.

[47] Hirak Sarkar, Mohsen Zakeri, Laraib Malik, and Rob Patro. Towards selective-alignment: Bridging the accuracy gap between alignment-based and alignment-free transcript quantification. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '18, page 27–36, New York, NY, USA, 2018. Association for Computing Machinery.

[48] Bo Li, Victor Ruotti, Ron M Stewart, James A Thomson, and Colin N Dewey. Rna-seq gene expression estimation with read mapping uncertainty. *Bioinformatics*, 26(4):493–500, 2010.

[49] Christopher Lee, Catherine Grasso, and Mark F Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, 2002.

[50] Heng Li. Wgsim.

[51] Valentine Svensson. Droplet scRNA-seq is not zero-inflated. *bioRxiv*, page 582064, 2019.

[52] Dominic Grün, Lennart Kester, and Alexander Van Oudenaarden. Validation of noise models for single-cell transcriptomics. *Nature Methods*, 11(6):637, 2014.

[53] Peijie Lin, Michael Troup, and Joshua WK Ho. Cidr: Ultrafast and accurate clustering through imputation for single-cell rna-seq data. *Genome Biology*, 18(1):59, 2017.

[54] Emma Pierson and Christopher Yau. Zifa: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome Biology*, 16(1):241, 2015.

[55] David Van Dijk, Roshan Sharma, Juozas Nainys, Kristina Yim, Pooja Kathail, Ambrose J Carr, Cassandra Burdziak, Kevin R Moon, Christine L Chaffer, Diwakar Pattabiraman, et al. Recovering gene interactions from single-cell data using data diffusion. *Cell*, 174(3):716–729, 2018.

[56] Mo Huang, Jingshu Wang, Eduardo Torre, Hannah Dueck, Sydney Shaffer, Roberto Bonasio, John I Murray, Arjun Raj, Mingyao Li, and Nancy R Zhang. Saver: gene expression recovery for single-cell rna sequencing. *Nature Methods*, 15(7):539, 2018.

[57] Wei Vivian Li and Jingyi Jessica Li. An accurate and robust imputation method scimpute for single-cell rna-seq data. *Nature Communications*, 9(1):997, 2018.

[58] Cédric Arisdakessian, Olivier Poirion, Breck Yunits, Xun Zhu, and Lana X Garmire. Deepimpute: an accurate, fast, and scalable deep neural network method to impute single-cell rna-seq data. *Genome Biology*, 20(1):1–14, 2019.

[59] Divyanshu Talwar, Aanchal Mongia, Debarka Sengupta, and Angshul Majumdar. Autoimpute: Autoencoder based imputation of single-cell rna-seq data. *Scientific Reports*, 8(1):16329, 2018.

[60] Romain Lopez, Jeffrey Regier, Michael B Cole, Michael I Jordan, and Nir Yosef. Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12):1053, 2018.

[61] Gökcen Eraslan, Lukas M Simon, Maria Mircea, Nikola S Mueller, and Fabian J Theis. Single-cell rna-seq denoising using a deep count autoencoder. *Nature Communications*, 10(1):390, 2019.

[62] Luke Zappia, Belinda Phipson, and Alicia Oshlack. Splatter: simulation of single-cell rna sequencing data. *Genome Biology*, 18(1):174, 2017.

[63] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[64] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[65] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[66] Mohamed Marouf, Pierre Machart, Daniel Sumner Sumner Magruder, Vikas Bansal, Christoph Kilian, Christian F Krebs, and Stefan Bonn. Realistic in silico generation and augmentation of single cell rna-seq data using generative adversarial neural networks. *bioRxiv*, page 390153, 2018.

[67] Arsham Ghahramani, Fiona M Watt, and Nicholas M Luscombe. Generative adversarial networks simulate gene expression and predict perturbations in single cells. *BioRxiv*, 2018.

[68] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. Gain: Missing data imputation using generative adversarial nets. *arXiv preprint arXiv:1806.02920*, 2018.

[69] Grace XY Zheng, Jessica M Terry, Phillip Belgrader, Paul Ryvkin, Zachary W Bent, Ryan Wilson, Solongo B Ziraldo, Tobias D Wheeler, Geoff P McDermott, Junjie Zhu, et al. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8:14049, 2017.

[70] Güllü Görgün, Tobias AW Holderried, David Zahrieh, Donna Neuberg, and John G Gribben. Chronic lymphocytic leukemia cells induce changes in gene expression of cd4 and cd8 t cells. *The Journal of Clinical Investigation*, 115(7):1797–1805, 2005.

[71] Ashish Bora, Eric Price, and Alexandros G Dimakis. Ambientgan: Generative models from lossy measurements. In *International Conference on Learning Representations (ICLR)*, 2018.